(1)

INDRA
Technical
Report 48

UNIVERSITY COLLEGE LONDON
ARPANET PROJECT

ANNUAL REPORT

1 JANUARY 1977 - 31 DECEMBER 1977

PROFESSOR PETER T. KIRSTEIN

APRIL 1978

Submitted to:

Defence Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington
Virginia 22209

Attn. Dr. Vinton G. Cerf

This research was partially supported by the US Office of Naval Research under Contract N00014-77-G-0005

Dept of Statistics & Computer Science
University College, London

TABLE OF CONTENTS

ABSTRACT

This report describes the work of the UCL INDRA group during 1977. The bulk of the report consists of papers which have been published or are in the press. These cover the group's activities with the UK Experimental Packet Switched Service, with measurement tools for a Packet Satellite Network, with measurement of a specific Host-Host Protocol, and with the combination of Facsimile Techniques with Message Processing in packet-switched data networks. There are also short summaries of the UCL work on the X25 Network Access Protocols and on the interconnection of computer networks. Finally, the current usage being made of the UCL node of ARPANET is analysed.

I INTRODUCTION

Each year the INternational Display and Remote Access group (INDRA) at University College, London (UCL) produces an annual report. This report is designed to summarize our work, and is in any case a contractual requirement of many of the organizations supporting the work. This annual report is extremely costly in resources to write, but is a valuable exercise focus in forcing us to present our accomplishments.

In mid-1975, we presented a series of seven programs at conferences. These were bound together as a Technical Report (INDRA 1975). The 1975 annual report was based heavily on that reference. This year we have written five papers near the end of 1977, and also published a number of INDRA reports. For this reason, I have decided to base this annual report principally upon these published papers.

One disadvantage of this approach is that it does not provide uniform coverage of those projects we are pursuing. The length of the chapters is only a reflection of the papers presented, not an evaluation of the importance of the work. Our work with EPSS (chapter 3), SATNET (Chapter 5), Measurements (Chapter 6) and Facsimile (Chapter 7) represent mature activities, and therefore are covered adequately by this approach. Two projects, the multi-machine system (Chapter 2) and the analysis of ARPANET usage (Chapter 8), are so mature, that the work is more in the nature of service activities than research. The research aspects have been treated in earlier annual reports and papers. No papers on this subject have been prepared this year, so that short progress reports are provided. The publications of the group are listed in Chapter 9.

We have other important research activities - one is on problems with the interconnection of networks, another work with the X25 Network Access Protocol. Some of our work includes also connection of networks with X25 access protocols. The work on the interconnection of networks has progressed substantially during the year, but not in a sufficiently definitive way to produce a paper. We have decided, therefore, to pass over these aspects of the work for this report -- except where it is alluded to in Chapters 2, 3, 4 and 5. Our work on simulation of network protocols has continued, but has not been digested sufficiently to summarize here. Therefore this work will not be reported. A project on a Network Access Machine has been completed, and a very comprehensive thesis produced (KENT 1977). This activity was independent of others, and is heavily dependent on the PDP9. It has been applied, reasonably successfully, to automating some of the Facsimile Control of Chapter 7. This activity will also not be described further here, but is described in the reference.

A major new area into which we are putting considerable effort is our work with the X25 Access Protocol. This work is too new to have reportable results, but represents a sufficient fraction of our research effort that we must describe both our intentions and our progress. This work is discussed in Chapter 4.

Finally, we must acknowledge gratefully support from a number of organisations for whom this report represents the annual report. These organisations are the Atomic Energy Authority (Agreements MIC 69324 and AGMT/CUL/936), the British Library (SI/G/093 and 172), the Ministry of Defence (AT/2047/064), the Science Research Council (B/RG/5981 and 67022) and the US Defence Advanced Research Projects Agency via the US Office of Naval Research (N00014-77-B-0005).

II THE MULTIMACHINE SYSTEM AND NETWORK INTERCONNECTION

2.1 The Multimachine System

During 1977, progress with this system was slower than anticipated. This was partly due to the software complexity, and partly due to the change in personnel on the project. However, substantial progress was made, and we were in a position to prepare to provide a multi-system in 1978.

The release of the basic SWITCH system, described fully in previous reports, (e.g. KIRSTEIN 1977) was made in late 1976. This provided a basic mechanism for attaching different Hosts and Networks. All the newer subsystems written since mid 1976 were written in a form to fit under SWITCH. These included the ULCC CDC system, EPSS and the Culham GEC 4080 system. All three of these became operational for terminal traffic by the middle of 1977. An experimental service of the Culham GEC 4080 and EPSS was run together for a few hours a day from mid 1977. The system has operated reasonably reliably. Our experience with the EPSS portion is discussed in Chapter 3. The interactive facilities for the Rutherford (RL) IBM360/195 system was also brought under SWITCH in early 1977. The combined RL, ULCC and RSRE systems were run together experimentally for a few hours a day for several weeks. This service was also fairly reliable, but was soon withdrawn. The use of the RL system is so well established, that file transfer is an absolute requirement. For this reason we decided to withdraw the CDC/RL service under SWITCH until file transfer facilities could be supported also.

The file transfer capability for the ULCC system was provided early in the summer, and was adequate, but needed two improvements. First, the log-in was implicit, so that no general service could be offered. Second, it ran with an operating system release at ULCC which was being replaced. Making the total of SWITCH/CDC system operational (supporting interactive terminal and file transfer under SWITCH), was largely completed by the end of 1977.

The provision of file transfer capability under SWITCH was needed for all these systems. Since this required some basic systems work, we decided to concentrate first on making the ARPA-CDC portions work. After this we concentrated on the IBM 360 portions. By the end of 1977, the RL System under SWITCH (with interactive terminals and file transfer) was largely operational. We expect to integrate progressively the ARPA, RL, ULCC, RSRE, CULHAM and EPSS system together during the first two quarters of 1978.

Our initial feeling is that although the basic PDP 9 is a swapping virtual system, too much code will need to be resident to envisage putting all the systems together on one machine. We would like to get into a position to have ARPANET, RL, ULCC, RSRE and the slow EPSS line on one PDP 9 computer, with ARPANET, the fast EPSS line and Culham on the other. Whether this will be possible remains to be seen.

In general we expect to move into a situation where one PDP 9 will be a service machine almost all the time, and the other one all but perhaps 4 hours/day. Moreover, increasingly all service traffic should come over EPSS and not into the TIP directly. We have several motivations for going in this direction. First, the PDP 9s are reaching the end of a useful research life; they have a sophisticated and useful set of software concerned with current networks and systems, and this should be used as fully as possible. Secondly, with the emergence of commercial trans-atlantic services, such as the International Packet Switched Service, it will become unacceptable to the PO for us to operate private shared network services, over which they have no control at all. By passing through a public (albeit experimental) carrier network like EPSS, the PO can keep some control of the usage; the direct terminal access may then become unnecessary, so that we would not be crippled if it was forbidden.

2.2 Interconnection of computer networks

Over the year much work has been done both on the overall architecture of connecting computer networks, and also on specific problems. We have been able to draw substantially on our experiences with the SWITCH gateway mentioned above and our initial experiences with the gateway between ARPANET and SATNET of Chapter 5. The two gateways represent almost two extremes in approaches in connecting different networks; thus we have been forced to consider very broad approaches to interconnection which have served us in good stead for current activities; this work will provide a basis for future work more related to X25 networks, and the connection of local to national facilities.

Despite the advent of X25 the interconnection of datagram networks is a major topic of interest both for the connection of local networks - especially where ring, or broadcast nets are involved, and also in defence networks where reliable communication and the use of networks operating over very different media, make the X25 approach unsatisfactory. The drive to obtain totally acceptable protocol structures to support this form of intercommunication, irrespective of short timescales, makes the subject a very different one from X25 oriented work. In the latter, compromises have to made in order to achieve service objectives in the remaining years of the 1970s.

Contributions in this wider environment are very different from the rest of the group's work; it has not been easy to make useful advances with the very limited manpower available on the project. The group's main contribution during 1977 was the production of a comprehensive position document (BENNETT 1977), which was discussed and distributed widely among the ARPA Internet Working Group studying this problem area. The production of the position document enabled us to identify particular areas in which we felt able to concentrate our resources. We defined problem areas as follows:

addressing

flow control

routing

gateway maintenance

gateway fragmentation

the requirements for an internetwork protocol

In particular we have clarified the existing understandings on addressing, and we put forward novel ideas on gateway fragmentation, gateway flow control, and gateway maintenance. In this work, we are using results obtained from the TCP simulation work mentioned elswhere. The existence of the ARPANET/SATNET gateway on which the GNOME measurement tool resides, (Chapter 5) enabled us to study at first-hand the control of a gateway from a remote location, and the obvious flow control and congestion problems of datagram gateways without built-in control procedures and conventions. We have also had the opportunity to study possible modes of operation for the interconnection of a datagram network with an X25 mode of operation.

During the latter half of 1977, the provisional CCITT recommendation was produced for the connection of X25 Virtual Call Networks. The group participated in the formulation of an IFIP submission to CCITT; in 1978 we expect to utilise more fully the experience gained on internetworking to comment further on CCITT plans.

The need to support an X25 DTE interface at the connection point between private and public networks has considerably altered the likely nature of future interconnection approaches. The UCL SWITCH gateway represents a service gateway approach that will be increasingly adopted where protocols are incompatible and no end-to-end addressing is feasible. The British Universities plans for an X25 network are an ideal forum for the development of these, and during late 1977 the group started to become involved in the discussions about the evolvement of the different interconnection approaches which will be taken by different regional UK networks. Out of our previous

work, the concept of a service gateway is likely to be extended and to some extent formalised into the different levels of mapping that may be required to be performed in order to provide certain services.

III ACTIVITIES WITH EPSS

Our activities and experience with EPSS are discussed in a paper which will be presented at EUROCOMP 78 in May 1978. Rather thar repeat this description, this paper is presented in its entirety in this Chapter.

PAPER 4

EXPERIENCES WITH THE INITIAL EPSS SERVICE

P.L. HIGGINSON and Z.Z. FISHER

Department of Statistics and Computer Science
University College London

Abstract

A minicomputer at University College London has been connected as a Gateway to both the EPSS and the ARPA packet-switching networks, so that users on one network may access resources on the other. An overview is given of the Gateway system and the structure, implementation and operation of the EPSS part of the connection is presented in detail. Progress to date and planned future activities are reviewed, and the relevance of this work to future public networks is described.

1. INTRODUCTION

The Networks Research Group at University College London (INDRA) has had a connection to the network of the Advanced Research Projects Agency (ARPA) for many years, Recently the group has connected its PDP-9 mini-computers to the Experimental Packet Switched Service (EPSS) of the British Post Office. The original objects for making this connection were to gain experience with and assist in the evaluation of EPSS, and to make it possible for users on the ARPA network to access EPSS and vice-versa. Recently we have come to believe we can also use EPSS to gain experience for the next generation of X25 networks.

This paper reports on our experience of making the connection to EPSS, the problems encountered and describes the interface to the ARPA network software. It mentions briefly our future activities.

2. LEVELS OF EPSS PROTOCOL

The protocols in use on EPSS are structured as a number of independent layered protocols with each protocol using the services provided by the level below and providing services to the level above. Figure 1 shows this diagrammatically.

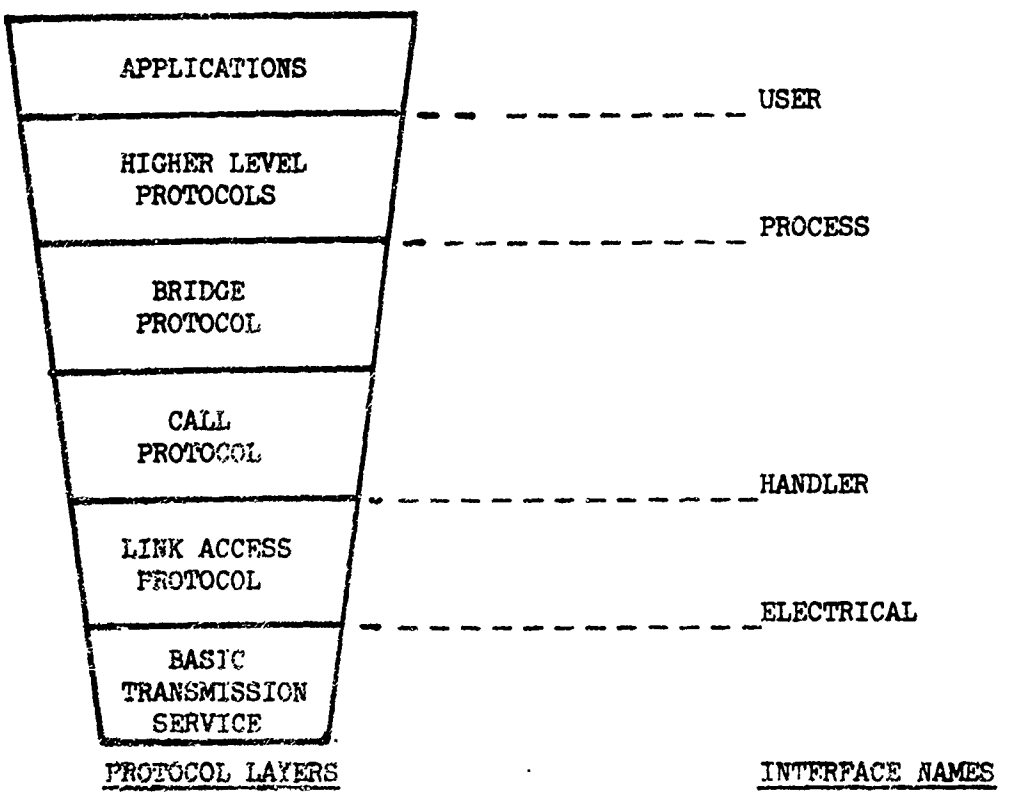


Fig.1. EPSS Protocols and Interfaces in our Implementation

The Link Access Protocol (Ref.1) uses a basic transmission service and provides a reliable ordered packet delivery service between the Call Protocol modules in the host computer and the Packet Switching Exchanges. The Call Protocol (Ref.1) utilises this reliable link to setup and support a number of independent data streams between various host computers. The Bridge (Ref.2) enhances this with extra control facilities both for establishing and clearing the links and for better flow control. Figure 2 shows how these protocols fit into our implementation.

At the higher (Process) interface to Bridge the data has been demultiplexed and subjected to controls, but has not yet been interpreted. The Higher Level Protocols give meaning to the data by converting between the formats and conventions used in the host computer and those standardised for the network. For example, for terminal access to interactive application programs the higher level protocol might involve editing of input, and adding format effectors and padding to output; the exact details are host dependent. For file transfers, the File Transfer Protocol converts between standardised network conventions and the local file store access interfaces.

3. GENERAL STRUCTURE OF OUR GATEWAY IMPLEMENTATION

Figure 2 shows the general structure of the EPSS/ARPA connection within our gateway. The system we run also supports a connection to the Culham Laboratory which is not shown in the figure and is not relevant to this paper. The modules for the Culham system are linked in at the 'User Interface' and other systems can be similarly supported up to the capacity of the computer. For simplicity, the drivers which invoke the protocol modules, the local command software and the interfaces to the local teletype, punch and printer (for operator messages, statistics and selectable package monitoring)are not shown.

It is not intended in this paper to discuss the Switch System, which was outlined at an earlier Eurocomp (Ref.3), or the Arpanet connection modules, most of which have been operational for some time even though many changes were made (particularly to the File Transfer module) when the system was interfaced to Switch. However, in order to put the EPSS modules into context, it is necessary to say a little about the facilities available.

The Switch System is a set of software modules within the PDP-9 which permits the dynamic establishment of links between streams at the User Interface, These streams are then mapped into calls in the appropriate protocol by the EPSS or Arpanet modules. Switch also provides common facilities for commands, status and help to interactive users from any network.

Requests to Switch fall into four basic categories - Control, Service, Data and FTP. Control requests to allow the allocation of queues to new users and tidying up when users go away. It is also

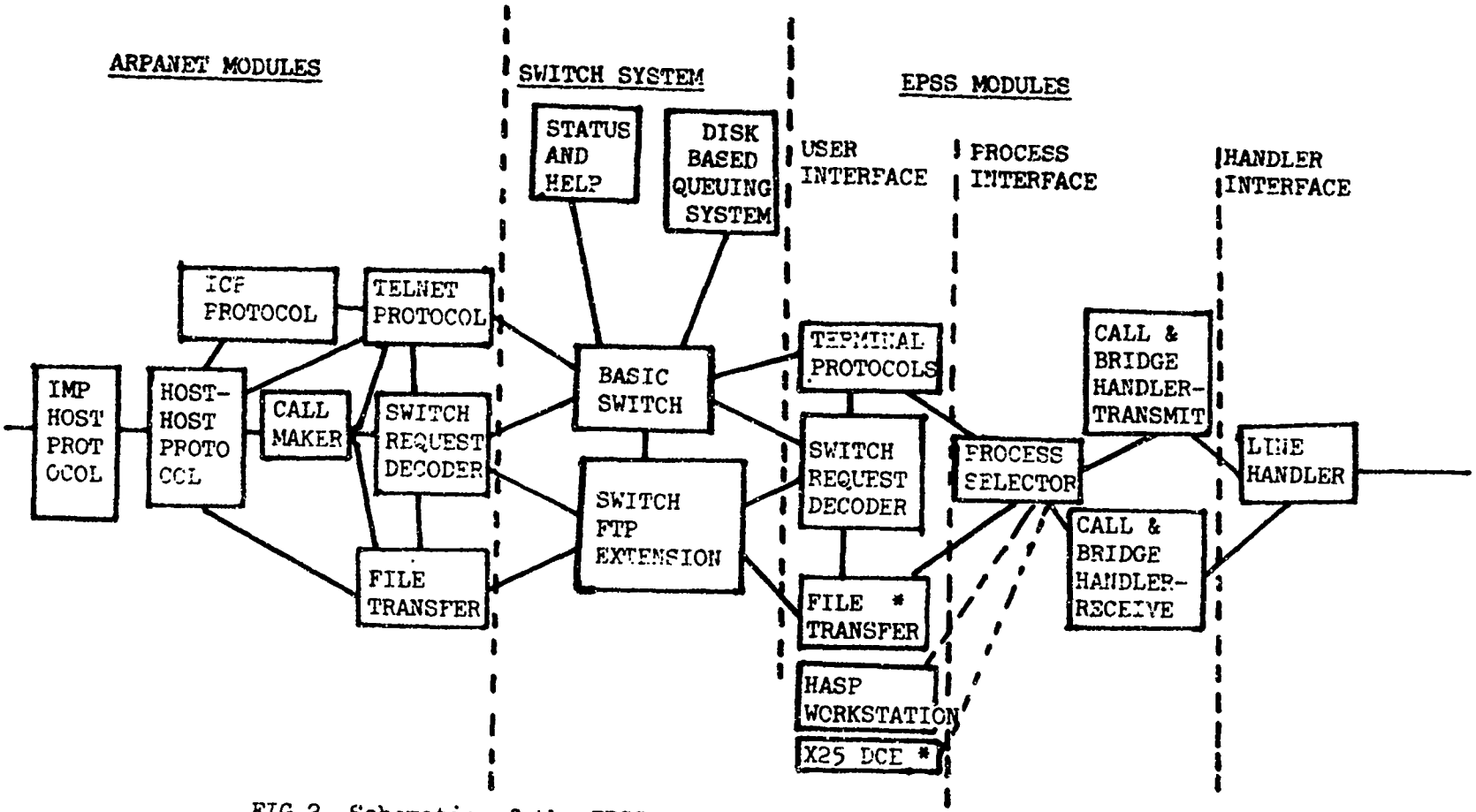


FIG.2 Schematic of the EPSS-ARPANET CONNECTIONS IN THE GATEWAY

* not yet operational

possible during a call to request its status and the amount of data queued. Service requests can be typed by interactive users and govern the setting up and clearing of connections to other networks; these requests are also used directly by programs. Data requests are used to forward data across the connection, send data back to a user and pick up data queued for a user. File transfers are organised by using the Control and Service requests to setup a link between two file transfer modules, the FTP requests to initiate, accept and terminate the file transfer and the Data requests to pass over the file contents.

The Arpanet modules have the capability to accept calls fron interactive terminal users on Arpanet, to make interactive calls (on behalf of users on other networks via Switch) into Arpanet, and to transfer files in either direction at the request of users on Arpanet hosts. The Arpanet modules to allow other networks to control file transfers are not yet operational and hence the user is required to make his transfer request to an Arpanet host. Switch and other file transfer modules (e.g. the Culham one) can support control of transfers by either party.

4. OVERVIEW OF OUR EPSS INTERFACE SOFTWARE

Our EPSS software is divided into three sections by the User, Process and Handler interfaces as shown in Figs. 1 and 2. Following experiences with our Arpanet implementation, where the levels of protocol are more mixed and our implementation less well structured than shown in Fig.2, we have attempted to provide rigid separation at the three interfaces.

At the handler interface, a flow of packets passes in each direction. All the packets passed down through this interface are sent in the same order to the PSE and all packets correctly received are acknowledged (if necessary) and passed up to the call handler.

The Call and Bridge Handler demultiplexes the individual data streams and passes the received data via the process selector for decoding by the high level protocol routines. In the reverse direction, data prepared by the high level protocol routines is given to the process selector for transmission. As soon as flow control permits and local buffers are available, the call handler puts the data into packets and passes them to the line handler.

A virtual call service with reliable delivery and flow control is provided to 'processes' which run above the Process Interface. We currently run a decoder for the Character Virtual Packet Terminal (ChVPT) protocol which is the protocol used by character terminals which have dialled up the EPSS Packet Switching Exchange directly (appendix E of Ref.1) and by character terminals on hosts which wish to access interactive services. Input and output for character terminals is converted between the ChVPT format and the internal format used by Switch. On the far side of Switch the Telnet Protocol (Ref.4)

program does the equivalent for the Arpanet.

We are in the process of setting up a File Transfer process which will convert between the EPSS File Transfer Protocol (Ref.5) primitives and the primitives used across Switch (Refs. 6 and 7). One of the Arpanet modules does the equivalent for the Arpanet File Transfer(Ref.4).

5. THE PROCESS INTERFACE AND NON-STANDARD PROCESSES

The process interface allows any number of high level protocols to be used above the virtual circuit service provided by the Call and Bridge Handler. The primitives of this interface are listed in Annex 1 and it is worth noting that they follow closely with the guidelines laid down in the Bridging Ptotocl Specification for the Higher Level Interface (Chapter 3 of Ref.2).

Apart from the terminal and file transfer protocols mentioned above, we have had a workstation interfaced to EPSS at the process interface and we plan to interface an X25 DCE in the same way. The process interface is also configured to support the 'datagram type' single packet calls of the EPSS. We expect to use this facility to pass some of the Internetwork protocols, (e.g. Ref.8) on end-to-end datagram services through the Gateway. Another possible process using this facility would be a statistics and availability routine using the Echo process on other EPSS hosts via single packet calls.

6. THE WORKSTATION PROCESS

In late 1976 we implemented a complete Rutherford Laboratory Workstation as a process connected to EPSS. The Rutherford Laboratory have extended the IBM HASP 1130 workstation to support interactive terminals in addition to the normal card readers and line printers, and we had an implementation of this which mapped the workstation streams onto Arpanet. We modified this system by altering the code which gave a packet to the line hardware for transmission, to give it to the Process Selector instead and to take a packet from the Process Selector and pretend it had received it from a telecommunications line. We arranged that, instead of sending the initial 'ENQ' message, it opened a call, and instead of sending the final 'SIGNOFF' message, closed it. The CRC check was also disabled in the Workstation software, since the EPSS packets had their own CRC check.

The system then worked as a workstation and used one EPSS call for all its traffic. We ran it in parallel with a leased line version (on a different machine) and the users could not distinguish the two systems easily. The commands to the two systems were identical, of course, and the faster line speed through EPSS made up for the three hops (UCL-PSE-PSE-RL) and the smaller packets. In fact, whether outputs came in bursts of six lines or four was the only easy way to tell which system was being used without examining some physical aspect of

the connection. Some of the conclusions which were drawn from this experiment are discussed in Section 23.

7. THE X25 DCE PROCESS

We have under development a process which will convert between the process interface and the CCITT X25 Level 3 protocol (Ref.9) as would be implemented by a packet switching service. The complementary Level 2 implementation and HDLC hardware interface are undergoing testing at the moment. This system will be used to provide a realistic X25 exchange with calls setup through EPSS as well as some possible interworking of our X25 terminals with EPSS.

8. FILE TRANSFER MODULE

The primitives of the Switch FTP for requesting file transfers are based on the EPSS File Transfer Protocol, but the formats for the messages were changed so that they were easier for programs to decode. A parameter buffer is built by one process and decoded (usually without being moved) by another. For the data phase, the existing queuing system is used for the data transfer, but the direct control remains for the passing of control signals.

9. CHARACTER TERMINAL PROTOCOLS

The simple transparent ChVPT protocol defined by the Post Office has been implemented by most of the active EPSS users. It is a simple, completely transparent, teletype compatible protocol and it is relatively easy for hosts which wish to make outgoing calls to mimic enough of the protocol to make a host-host connection work. A need for standardisation of the use of the ChVPT protocol by hosts has recently become apparent in order to avoid host implementations being incompatible with one another. This experience shows that there is a need for great comonality in the use of terminal protocols by hosts, if terminals interfaced to local networks and directly to hosts are to use the same protocols as character terminals interfaced directly by the network.

10. OUR IMPLEMENTATION OF THE CHARACTER VIRTUAL PACKET TERMINAL

Because the ChVPT is a transparent protocol, the host must provide options for the support of different types of character terminal. We provide support for three types of terminal and the type of support selected is decided by the Process Number selected by the caller. The Process Number is a Sub-address given by the user and it is used to select the protocol, the service and any local options; in some cases it can be extended into the data part of the call originating packet.

Two of the types of terminal supported are intended for VDU's and printing terminals connected to the PSE. In these cases, if the user types a line terminated by carriage return, a line feed is sent back to complete the new line. The third type is intended for terminals on hosts where the line feed will be provided locally and not sent back across EPSS. In addition, for the PSE terminals, input may be terminated by DEL in order to abort a line being input and by Control-N in order to forward a partial line. This interpretation is carried out by our host software, not by the PSE. For printer terminals, padding characters are inserted in the output.

We do not at the moment provide either variable amounts of padding or for parity generation or checking. The advantage of a transparent protocol is that these can be provided should we wish. The disadvantages are the cost of all hosts having to provide support for several types of terminals and the asymmetry that usually results, whereby certain terminals are only usable with certain hosts. We also know from experience that some standardisation of usage is necessary to permit host-host working.

11. THE CALL AND BRIDGE HANDLER

The Receive and Transmit handlers are relatively straightforward and deal with flow control, multiplexing and demultiplexing of the packet stream and call setup and clearing. So far we have made the assumption that EPSS is reliable and not implemented packet retry or complex error handling. If errors do occur we use the reset mechanism to get a call back to a known state. We assume that interactive users will do their own recovery and that the restart mark facilities of the File Transfer Protocol will be used to recover errors in bulk transfers.

In a paper of this size a detailed description of the call handling software would be inappropriate. In general, we feel the software is more complex because of the number of error conditions in EPSS (i.e. the number of Network Information Packets) which is higher than X25, for example, where errors automatically reset or clear the call and the host has no other options (compare Refs. 1 and 9).

12. BUFFERING STRATEGY

The strategy for managing local core buffers is very important if copying of packets around core is to be minimised. The strategy developed for EPSS has been used in our X25 implementations and we feel it is the best strategy for a small computer. Obviously if there were a pool of system managed buffers, then it would be profitable to use it, but there are problems due to the need to add or remove headers from the start of packets as they are processed by the call handler.

The Line Handler owns and manages a small (currently 4) number of packet buffers which are used for both transmission and reception.

When the start of a packet from the PSE is detected a buffer is found and the packet read into it. After checking, this buffer is queued for the Call and Bridge Handler. The Call and Bridge Handler is given one received packet at once, and must deal with it immediately. On returning the buffer to the Line Handler it is given the next received packet if any.

The Call and Bridge Handler processes the packet header and passes the data part of the packet via the Process Selector to the appropriate higher level protocol process; again this process must deal with the data immediately, normally by writing it into the Queuing System via Switch. In this manner, the process has no knowledge of the EPSS packet header and we avoid copying the data into a process area unless the process has to copy it as part of its processing of the data.

For transmission the Call and Bridge Handler requests a buffer from the Line Handler and if one is available, attempts to fill it with a packet header and with data supplied by a process. When filled the buffer is sent to the Line Handler for transmission, and when transmitted (and acknowledged if necessary) the buffer is freed. In some cases the process prepares a block of data in its own buffer which is then queued by the Process Selector until a packet buffer is available and flow control permits transmission. For efficiency purposes, processes which forward data directly from Switch queues sub-contract their data preparation to the Process Selector. This means that the queues (which are disc based) are not read until a packet buffer is available, and avoids the output data waiting in core for EPSS flow control to permit its transmission. When handling a large number of calls this excess core usage would be intolerable in a small machine. The sub-contracting involves for terminal streams, for example, the inserting of line feeds and padding into output.

13. THE LINE HANDLER

The Line Handler is written to be able to use either the Standard EPSS Link Access Protocol through a Transmission Protocol Unit (TPU) purchased commercially or the simplified Link Access Protocol. It also can use either a slow speed synchronous line adaptor or a specially modified one for working at 48K bps. The handler currently initialises to one of the four modes but could easily be extended to determine the mode at each line break if we required this.

We have two lines to EPSS, one at 48K bps using the Standard Protocol and another at 2.4K bps using the Simplified Protocol. We have a TPU which handles retransmission, CRC generation and error checking for the Standard Protocol and we use this on the 48K bps line. We intend to allow use of the Simplified Protocol on this line as a fallback should the TPU fail, but although we did some tests on the Handler we have never pursued this. The Simplified Protocol has less critical timings and does not interleave acknowledgements with packets being sent. This means that the Simplified Protocol is easier to

handle directly by software although the maximum achievable throughput is less than in the standard case. How much less depends on the type of traffic: for half duplex traffic or for an unsaturated line (say less than 70% loaded) there will be little difference. For a line saturated in one direction by large packets, the reverse direction will be restricted severely if it is sending small packets.

The handling of the start of a packet caused problems at 48K bps since the length of a packet has to be determined from the first two bytes in order to calculate the size of the data channel transfer request. The inter-character gap was too small to permit interrupt handling by our software and hence our hardware was modified to both interrupt and start storing the characters in core. The software then picks up as many characters as have been stored by the time it is called (the problem is mainly the length of time other routines disable interrupts).

The CRC calculations for Simplified are lengthy enough that they are done at user level within our system, a reasonable amount of time is available for the calculations.

14. THE UCL SERVICE OFFERING

Since April 1977 we have been running a service whereby EPSS users can access Arpanet. This has been made available to a few of the UK users of Arpanet who would normally have had to dial-up the TIP in London to obtain access. Due to lack of our resources the service has only been run in the morning (normally 8.45 to 11.00) but it is hoped to be able to extend the hours of running soon.

We have not increased the number of users partly because of the hours of availability and partly because experience has suggested that a number of improvements are necessary. These improvements are discussed in sections 17 and 18. In addition, a premature attempt to support a small experiment on conferencing via dial-up to EPSS in March-April 1977 has made us cautious in our attempts at usage.

There was no single identifiable cause for the failure to support this experiment over EPSS. Both our PDP-9 and the EPSS software was unreliable at this stage and has since been improved; obscure hardware errors were discovered in EPSS (call reallocation when one PSE was down) and our PDP-9 (the interface was sensitive to certain message lengths). Another problem is the difficulty of connecting through EPSS and is described in 18., however, at this time we disconnected the EPSS user whenever the Arpanet call was cleared and this meant that the problems were increased.

We have recently been putting effort into making EPSS hosts available to Arpanet and improving the characteristics of the service already available. We currently have a low but continuous level of usage and we would expect this to increase when we provide an improved

service with longer availability.

15. STATISTICS COLLECTED AND AVAILABILITY

The system monitors a reasonable amount of data on connections, messages and errors. Only since September, when we changed the monitoring to always output times and dates with messages, have we been able to assess performance accurately. Five weeks statistics, manually abstracted from this monitoring are given in Annex 2. Of the 25 days shown, a hardware failure kept the machine down on two, and collection problems gave incomplete statistics on one.

We have given the figures for total calls and usage, and a more detailed breakdown for the period 9.20 to 11.00. This second period was chosen to avoid the 'startup' of the EPSS service which appeared to often take EPSS down in the 8.30 to 9.00 period (even though we had had calls before 8.00 on many occasions) and to cover the time when we and EPSS were intending to provide a service. We thus give the number fo breaks in service for this period. For comparison we also give the availability figures for the period from 1am.

If we exclude the one day for which we have incomplete statistics, then we find our availablility during the 9.20 to 11.00 period is reasonably good and that EPSS was available for 93% of the time during which the system was running.

Since this paper was first prepared we have analysed the statistics for the following eight weeks. These are not reproduced because the call pattern is similar on days when the service was available, but due to various problems, on many days the service was not available. Out of 40 days, the service was available and connected to EPSS for only 22 days for the whole of the period 9.20 to 11.00, the remainder being accounted for as follows:

TPU problems	5 days	
Power cuts	3 days	(part service on 2)
EPSS not available	9 days	(part service on 1)
PDP-9 hardware	1 day	

This brings EPSS's availability relative to us down to about 70% for this period. This poor performance was mainly due to problems with our line card at the PSE and its effects were compounded by the necessity to wait until experts were available at our site in order to prove that the PSE was at fault and get the fault rectified. The TPU problems are not representative since the unit has otherwise functioned continuously over two years without failure.

16. USAGE

The statistics show a low amount of usage and virtually all of it

is accounted for by two users, the Royal Signals and Radar Establishment (RSRE) and the National Physical Laboratoey (NPL), both of which have computers attached to EPSS. There is little usage from users who dial EPSS directly and this is mainly because we have not offered users who currently dial up the London TIP directly the option of dialling EPSS instead.

There are several advantages in direct access to EPSS which do not apply to dial up users. Thus RSRE and NPL avoid line errors and have higher speeds of access than are available via either direct or EPSS dial up. Terminals connected to local facilities are often more easily available than terminals connected to modems and the computer can be used to save transactions for further processing. In addition, the users' overheads in setup time and cost are lower than the dial-up case since the connections already exist.

17. DIAL-UP USAGE THROUGH EPSS

Dial-up through EPSS is less attractive to users for four reasons. Firstly, the rather simple 6 character command to the TIP is replaced by 18 characters to EPSS and 24 characters to our Gateway (since both systems demand identification and extra information). Secondly, the system is less available and since the TIP is needed anyway, failures in the Gateway or in EPSS are additional to the other failures. Thirdly, although there is some cost difference, a telephone call is still necessary and many people regard their time as more important than their organisation's telephone bill. For users in the London telephone area, it will cost more to use EPSS when charges are made.

The fourth reason concerns the overall usability of the dial-up service. We regard this as poor at the moment and are trying to improve it. This involves both the connections through EPSS and the service in our Gateway.

18. DIAL-UP PROBLEMS THROUGH EPSS

The procedure for connection across EPSS is that the user dials a PSE in London, Manchester or Glasgow (whichever is the nearest) gets a banner and identifies himself (9 characters); he then types a connection request (9 characters) and hopes to receive a banner from the host. There is no way to correct these sequences and any failure results in disconnection of the telephone call. Two attempts are allowed at the identifier, but since a frequent cause of failure is noise pickup this causes the last character of the first attempt to be taken as the first character of the second attempt and ruins both. (This is particularly bad with acoustic couplers where inserting the handset often causes one noise character to be sent and this is taken as part of the password.) Experienced users leave the handset permanently in the acoustic coupler, type the identifier very slowly the first time and very quickly the second and never use an upper/lower case keyboard unless

it has a capitals only mode.

The problem with this connection procedure is that it discourages new users. Every time they make a mistake or take more than 30 seconds to complete the identifier or connection request, they have to redial. Faults of any type can also cause disconnections, as do unavailability of resources or host computers. One is simply not permitted to make a second attempt, even to call a different host, without redialling.

Having connected to UCL the user is relatively safe; we never disconnect him except when we restart after a crash (when immediate reconnection is possible). In any case, after a successful call which is cleared normally, another call is permitted. If the call hangs up, the user can either hope it is the network and attempt a reset (which will completely hang his terminal if it is UCL), or hope it is UCL and clear the call to avoid a redial (in which case he will unnecessarily lose work in progress if it is the network).

19. THE PROBLEMS OF MAPPING TERMINAL SERVICES IN THE GATEWAY

We have found that in order to provide a reasonable service we have had to support several types of terminal over EPSS, provide for forwarding of complete lines, incomplete lines and the throwing away of erroneous lines. Our Gateway totally restructures data in both directions using an intermediate format which is based on 7 bit characters and records of a line or part line. In the reverse direction a timeout is used to determine if an Arpanet host will complete a line.

Double echoing presents a problem which we have not attempted to solve. Most users need (and get) an immediate echo locally as they type a line, before it is forwarded across EPSS. Many Arpanet hosts expect to be doing the echoing and use this fact to alter what is echoed. To take a simple case, 'HI' typed by a user is echoed (by a sorting program called MSG) as 'Headers Inverse'. The user who turns off the Arpanet echo gets 'HI' on one line and ' eaders nverse' on the next, and the user who types 'HI <carriage return>' may well find that the <carriage return> has answered the programs next question. Hence our advice to users is to regard the second echo as confirmation that their input arrived in Arpanet and that it may be necessary to forward partial lines.

20. CURRENT GATEWAY INADEQUACIES

At the moment the Arpanet modules do not report loss of contact on calls and this is a limitation which we will rectify shortly. Calls which cannot be established are reported correctly and there are obviously cases in which the system does not know that contact has been lost. The queuing system has a 30000 character buffer and this is large enough to hold most terminal outputs. It is not large enough to hold all possible ones and certainly not large enough for file

transfers; hence we will have to implement some flow control in the EPSS Call Handler.

Reliability of the system is a matter of continuous improvement. We have very good facilities for automatic total restart (by software) in the event of errors or lack of progress by the system. Obviously changes to the sytem usually increase the frequency of restarts and careful monitoring and investigation of failures is necessary to bring the software reliability up to acceptable levels. We do not attempt to recover calls after a restart; in Arpanet the protocol does not allow for this and in EPSS it is complicated because the process originally called by the user cannot be identified. Hence we rely on users to reestablish calls.

21. FUTURE EPSS - ARPANET FACILITIES

We intend to provide for access to EPSS hosts by Arpanet users and for the transfer of files in both directions. We intend, if it proves reasonably possible, to permit the exchange of messages between EPSS and Arpanet. 'Mail' is highly developed in Arpanet with standard formats and many text processing programs to aid the construction, processing, distribution and answering of mail and we would investigate how this system could be extended so that EPSS and Arpanet addresses can be intermixed.

We have problems in providing general purpose connections (i.e. non-protocol specific ones) between Arpanet and EPSS because of the complexity of mapping all facilities of the respective call level protocols into each other. In particular, since Arpanet uses a separate stream for connection control, requests must be decoded with reference to the current call tables and one message may contain requests concerning several calls. This and other problems mean that the Gateway has to interpret parts of higher level protocols in order to make a connection work between the two networks.

22. TRANSNETWORK ADDRESSING

The Gateway has only one address in each of the networks to which it is connected. It is impossible for a user on Arpanet to request that a connection be setup via UCL and EPSS to NPL (for example) and hence the user makes one connection to UCL and then makes a second request for the EPSS-NPL part. We also take advantage of this second request to make the user identify himself for logging purposes. It would have been possible to use the EPSS process number to select an Arpanet host (since there are less than 256 hosts) but we would have lost the identification and we would still have had to provide the second command facility because some hosts have a number of services to which the user may connect.

File transfer and other automatic protocols pose more complicated

problems because the second command method cannot be used. For the Arpanet File Transfer Protocol we use a composite username field because the protocol is multi-request and the username is given in the first request. Thus to access a file in 'MY' account on NPL, I give 'EPSS.19290246.MY' or something similar. We intend both to allow a composit username field and to allow the user to give the Arpanet address in the EPSS Process field (as extended in Bridge) and wait to see which is the better.

The problem with automatic message protocols is very complex. Unless the message protocols evolve to include multi-network addressing, then the gateways between networks will have a large amount of context sensitive processing to do. For example, a user with a mailbox at Harwell will be known as 'USER at HARWELL' to mail systems on EPSS. On Arpanet we will construct the identifier 'USER/HARWELL/EPSS at LON-EPS-GATEWAY' to identify this user and route his mail to him. If Harwell has a local mail processing program then it will require a string such as 'USER2/ISI/ARPA at UCL' in order to send mail to another user whose Arapnet mailbox is 'USER2 at ISI'. The Gateway is going to have to convert the contents of the standard address fields (to, from, sender, reply to, cc, bcc) on all messages unless a format can be developed which avoids this.

23. RESULTS OF OUR EPSS WORK SO FAR

With many developments incomplete, it is difficult to quantify the results of our work so far. Experiences from the use of File Transfer Protocols and from serious production use of the Arpanet-EPSS link have yet to be evaluated. However, we can attempt to draw some conclusions from the work to date.

The interfacing of the Workstation Process described in Section 6 showed that the method of simply putting an existing data protocol inside the packets of a virtual call service was practical and very resilient to failures in the service. In X25 networks, where a Permanent Virtual Circuit can be used at the Workstation end, the amount of software conversion needed will be even less than in our case. For data packets, cost differences are marginal and in some cases better, because the workstation submultiplexes data on the call. We found high overheads due to the continual exchange of polling messages (ACKO in IBM HASP terms) and these would have to be removed in a production version. We intend shortly to experiment with a version which has the polling messages suppressed.

The experience we have of terminal protocols on EPSS shows a need for universal simplicity and uniformity in order to permit access by users to a wide range of hosts. (N.B. Our Arpanet experience shows the same trend). We are extremely worried by the complexity and lack of structure in the CCITT X3/X28/X29 protocols and fear that they will not be adequate for the purposes for which users will wish to use them.

In the area of network interconnection, we have established that it is possible to interconnect terminal traffic and have solved all of the problems which have arisen, as far as is possible. There do remain some problems, particularly the double echo discussed in Section 19, which reflect conceptual differences between EPSS and Arpanet. We have used the connection for real traffic between EPSS and Arpanet and this has been successful. We have had to adopt a double login procedure which is somewhat adhoc. It works reasonably well for two connected networks but would become impractical if several networks were transited between source and destination. A good solution to the general problem of transnet addressing has yet to be found.

The structure of our implementation has been used as a basis for our newer X25 implementations, and we feel we have found the right mixture of interfaces and self-contained modules to make a successful implementation of a network control program in a host.

24. FUTURE ACTIVITIES WITH EPSS AND X25 EXPERIMENTS

Development on networks is now concentrated on the higher level protocols and many of our current experiments are in this area. Now that X25 has been standardised as an access protocol, it is necessary to investigate the requirements for, and develop and test protocols for simple terminals, for data entry terminals, for access to file stores, for job exchange, for messages and documents, for facsimile and for workstations. Various EPSS groups or sites are working on most of these problems in order to use EPSS effectively and we are involved with many of them.

This work is network independent and is directly transportable from EPSS to an X25 public network. We are also developing the X25 DCE software (of section 7) to permit X25 terminals to access EPSS. These X25 terminals are being developed by us as part of the investigation of X25 as a network protocol, but their higher level protocols are independent of X25 and will be usable over EPSS and vice-versa.

25. CONCLUSIONS

The long gestation of EPSS is starting to show some dividends. The work on high level protocols is vital to the successful use of packet-switched networks and is independent of EPSS, but could and would not have been done without the stimulus of a network existing. This work can and will be transferred to the new X25 networks and a period of overlap between EPSS and the new X25 service is essential to allow a smooth transition. As a networks research group we are using EPSS as a vehicle for experiments in both call level and higher level protocols. The development of the Bridging Protocol in EPSS is important, because it is a network-independent interface for a virtual call network, and has allowed us to make a conversion to X25.

Not all lessons can be learned from EPSS by the middle of 1978 or even 1979; they are still being learned from ARPANET after nine years. It is of fundamental importance for the Post Office, the host interface builders and the users that they keep access to an operational network like EPSS. In some ways the fact that EPSS is 'experimental' allows freer access and more experimentation, without many of the technical and regulatory constraints that one may expect on a fully commercial successor. For most of these activities the non-standard protocol is irrelevant. For others, with the type of development mentioned in section 24, EPSS is suitable even to test X25 features. In spite of its slow start, if its operation is continued, we are confident that the lessons learned from EPSS will help to ensure that a success is made of an X25 service when one is provided. In this EPSS will have met one of its primary objectives as an experiment.

26. ACKNOWLEDGEMENTS

This work was supported by the Science Research Council under Grant B/RG/67022. It would not have been possible without the support of the Atomic Energy Authority under Agreement CUL/936 and ARPA under an Office of Naval Research Agreement N00014-77-G10005.

27. REFERENCES

1. Technical Guide No.16, The Experimental Packet Switched Service (Post Office Telecommunications Services).

2. Bridging Protocol Specification, EPSS Liaison Group, Study Group 3.

3. The Problems of Linking Several Networks with a Gateway Computer, P.L. Higginson, A.J. Hinchley, Eurocomp 1975.

4. Arpanet Protocol Handbook, ARPA Network Information Centre.

5. A Network Independent File Transfer Protocol, Higher Level Protocol Group

6. First Specification for a Switch FTP, P.L. Higginson, INDRA Note 580, Dept. Statistics and Computer Science, University College London.

7. U.C.L. Gateway File Transfer Interface, A.J. Hinchley, INDRA Note 607, Dept. of Statistics and Computer Science, University College London.

8. Specification of Internetwork Transmission Control Program. V. Cerf, Y. Dalal, C. Sunshine., INWG Note 72.

9. CCITT Recommendation X25, Geneva 1976.

ANNEX 1 THE PROCESS INTERFACE

1. Parameters of Calls

All calls have the following parameters:

Action	Word 0, bits 0-2
Type	Word 0, bits 4-5
Slot	Word 0, bits 6-17
Byte Count	Word 1, (-ve)
Data Address	Word 2

2. Actions

Most actions apply for all calls to and by the process:

Action	To Process	By Process
Data	Received	To Send
New Call	Received	To Open
Call Accept	Received	-
Call Close	Call Closed	Close Call
Data Fail	Reset Received	Send Reset

3. Type

The type parameter selects datagrams rather than virtual calls, whether transmission is direct or sub-contracted, and the half-word in which the data starts.

4. Slot

Slot identifies one of a number of possible EPSS connections. When a new call is opened by the process, the slot is returned (as a value) and is then used to identify this virtual call for subsequent actions.

5. Byte Count and Data Address

Give the size and address of data received or to be transmitted. A special format used for the 'New Call' action allows both the network address, the local and call facilities required and the process data field to be specified.

6. Return Values

In general, positive returns signal acceptance (OK) and negative returns, refusal or errors. For a 'New Call' by a process the return gives the slot number (which is positive).

Annex 2 STATISTICS FOR 19 SEPTEMBER to 21 OCTOBER 1977

Figures relate to the period 0100 to 1100 and are given for weekdays only.

Date	Time System Up(Hrs)	Time EPSS Up(Hrs)	Calls from EPSS	Calls to Arpanet	Successful* calls to Arpanet	Sum of call times (mins)
19 M	10	1.3	0	0	0	0
20 Tu	10	2	3	3	0	9
21 W	10	10	2	1	1	31
22 Th	10	10	9	7	6	146
23 F	10	10	0	0	0	0
26 M	10	10	15	12	8	92
27 Tu	10	10	2	2	2	123
28 W	no information					
29 Th	10	10	3	2	2	102
30 F	10	10	9	9	2	23
3 M	10	10	7	7	0	0
4 Tu	9	4.5	1	1	0	0
5 W	10	0.5	1	1	0	0
6 Th	10	10	11	11	5	124
7 F	10	10	19	13	1	16
10 M	Hardware failure					
11 Tu	"	"				
12 W	10	10	0	0	0	0
13 Th	10	10	2	1	0	0
14 F	10	10	1	1	0	0
17 M	10	10	2	2	0	0
18 Tu	10	10	3	3	1	79
19 W	10	10	4	3	2	49
20 Th	10	10	4	4	4	156
21 F	10	10	0	0	0	0

* 'Successful' was defined as lasting more than one minute. The difference is accounted for by hosts on Arpanet not being available, and by problems with the London TIP or the lines to the USA. The problems are exaggerated because users often made repeated attempts over short periods of time. In particular, the most used host is never available on Friday mornings.

Annex 2 Page II

More Detailed Breakdown of Statistics for period 9.20 to 11.00

Date	Time System Up(mins)	Time EPSS Up(mins)	No. of System Breaks	No. of EPSS Breaks	Calls to Arpanet
19 M	100	80	1	0	0
20 Tu	100	100	0	0	3
21 W	100	100	0	0	2
22 Th	100	100	1	0	7
23 F	100	30	0	1	0
26 M	100	100	0	0	6
27 Tu	100	100	0	0	1
28 W	no information				
29 Th	100	100	2	0	1
30 F	100	100	0	30	4
3 M	100	100	0	0	4
4 Tu	100	100	0	0	1
5 W	70	4	0	1	1
6 Th	100	100	1	0	5
7 F	100	100	0	1	10
10 M	Hardware Failure				
11 Tu	"	"			
12 W	100	100	1	0	0
13 Th	100	100	0	0	1
14 F	100	100	1	0	1
17 M	100	100	0	4	2
18 Tu	100	100	0	0	3
19 W	100	100	1	0	2
20 Th	100	100	1	0	4
21 F	99	99	1	0	0

IV ACTIVITIES IN THE CONTEXT OF THE X25 ACCESS PROTOCOL

4.1 Introduction

The X25 Network Access Protocol was approved by CCITT in 1976. It has become clear that all European, and some North American, public packet switched data networks will adopt this access protocol at least for the next few years. We have, therefore, started a substantial activity in this area. Our activity has many facets; it includes studying both the implications and possible variations of the X25 protocol itself, and the high level protocols being proposed above it. Key problems which we plan to study include the following:

(i) The problems in connecting X25 networks to others with different network access protocols.

(ii) The suitability of X25 as an access protocols for local networks.

(iii) The problems in connecting X25 networks together.

(iv) The testing of X25 networks.

(v) High level protocols above X25.

(vi) Methods of increasing the performance and reducing the size of X25 implementations.

(vii) Possible modifications of X25 in the light of the above.

Our current plans and activities are discussed in Section 4.2.

At this stage most of our activity has been to study how we can build up an experimental infrastructure. This requires Hosts, Terminals and Networks. The only concrete activities have been to start to develop a Test Network and two Experimental Test Hosts, which are described in Section 4.3 and 4.6. The status of systems at the end of 1977 is discussed in Section 4.4.

4.2 Overall UCL Plans

To be able to investigate the research items of Section 4.1, we need several X25 Networks and Hosts. Ideally we need at least the following system:

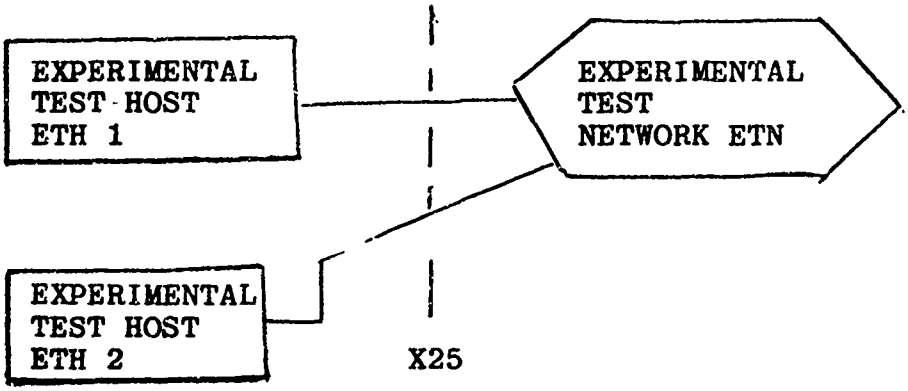


Fig 4.1 Two Hosts into an X25 Network

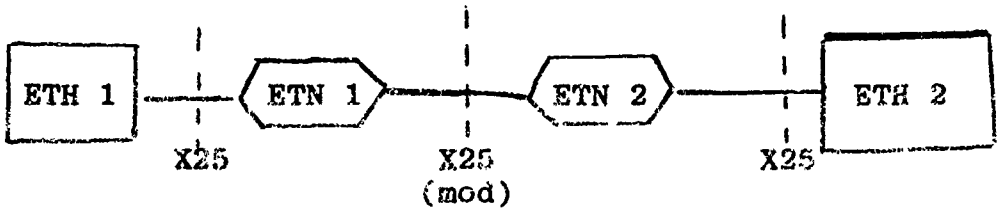


Fig 4.2 Two Hosts on Two X25 Networks

As Experimental Test Network we envisage, for the moment, the EPSS network of Chapter 3, suitably modified with a front end to support X25 access, and the SATNET of Chapter 5, suitably modified in the same way. Later we will probably use EURONET for some of our work as a real network - but it will not be accessible before early 1979. This report describes past work. During 1977 we have only considered how to make SATNET look like an X25 Network; no serious work has started in this area. For this reason we will only discuss the work with EPSS and the Test Hosts in this Chapter.

For the development of Fig 4.2, we envisage Test Hosts at COMSAT and UCL, and that the front ends of SATNET and EPSS (the Gateway PDP11 and the PDP9) act as half-gateways. The exact method of implementing this proposal has not been worked out fully.

4.3 Structure for the TEST HOST/TEST NETWORK Software

An overview of the way we are devloping an X25 front-end to EPSS and an Experimental Test Host is shown in Fig 4.3. Here all the software indicated on the right-hand side of the box labelled "PDP-9" was developed for our EPSS/ARPANET connections. The ARPANET software has been replaced, for the purpose of this work, by the X25-specific software (and hardware) on the left of that box. The box marked "LSI-11" will be the Experimental Test Host" (ETH). The form of connection of X25 to EPSS software in the PDP-9, being directly linked to the Call and Bridge Protocols sections, allows all high level protocols in the ETH to be passed through the network (in this case EPSS) transparently. This is different to the ARPANET-EPSS connection, where mapping of high level protocols takes place.

In Fig 4.3 the link between LSI-11 and the PDP-9 uses the standard LAP version of the X25 protocol, and can be broken to make either a host or a network port available to other users. A standard interface specification was developed for use at the boundaries between modules in the LSI-11; the same interface specification was used in the PDP-9 (above and below the X25 level 2) and in the INTEL 8080 for the Facsimile project. The box marked 'HDLC interface" includes both the hardware to drive the physical interface, and the lowest level software (called the Frame Handler) to convert between the standard interface and the hardware. This Frame Handler conceals, for example, the fact that the LSI-11 has a character interrupt interface whereas the PDP-9 has a packet interface using the data channel.

An equivalent box can be drawn for the INTEL 8080 Test Host. Here the main system has been developed for the facsimile project described in Chapter 7.

Previously the connection has been through the PDP-9 via another form of interface. Partly for increasing our experience, and partly because of the intrinsic value of the development, we are replacing the other communication interface by that of Fig 4.4. We aim eventually to have this system operate through EPSS (seen as an X25 Net); thus the types of usage of the two ETHs are essentially similar.

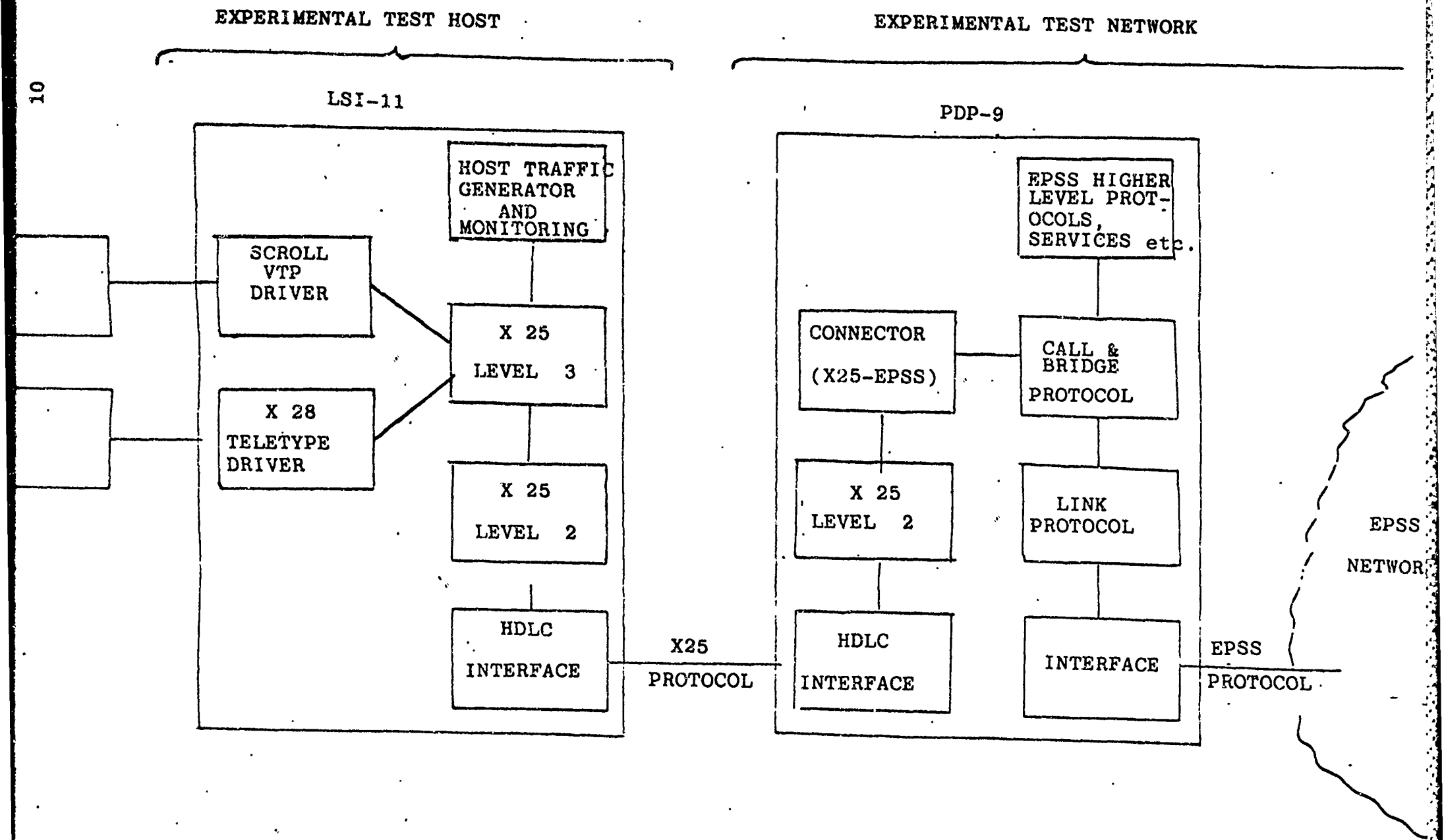


FIGURE 4.3 : STRUCTURE OF TEST HOST AND TEST NETWORK SOFTWARE

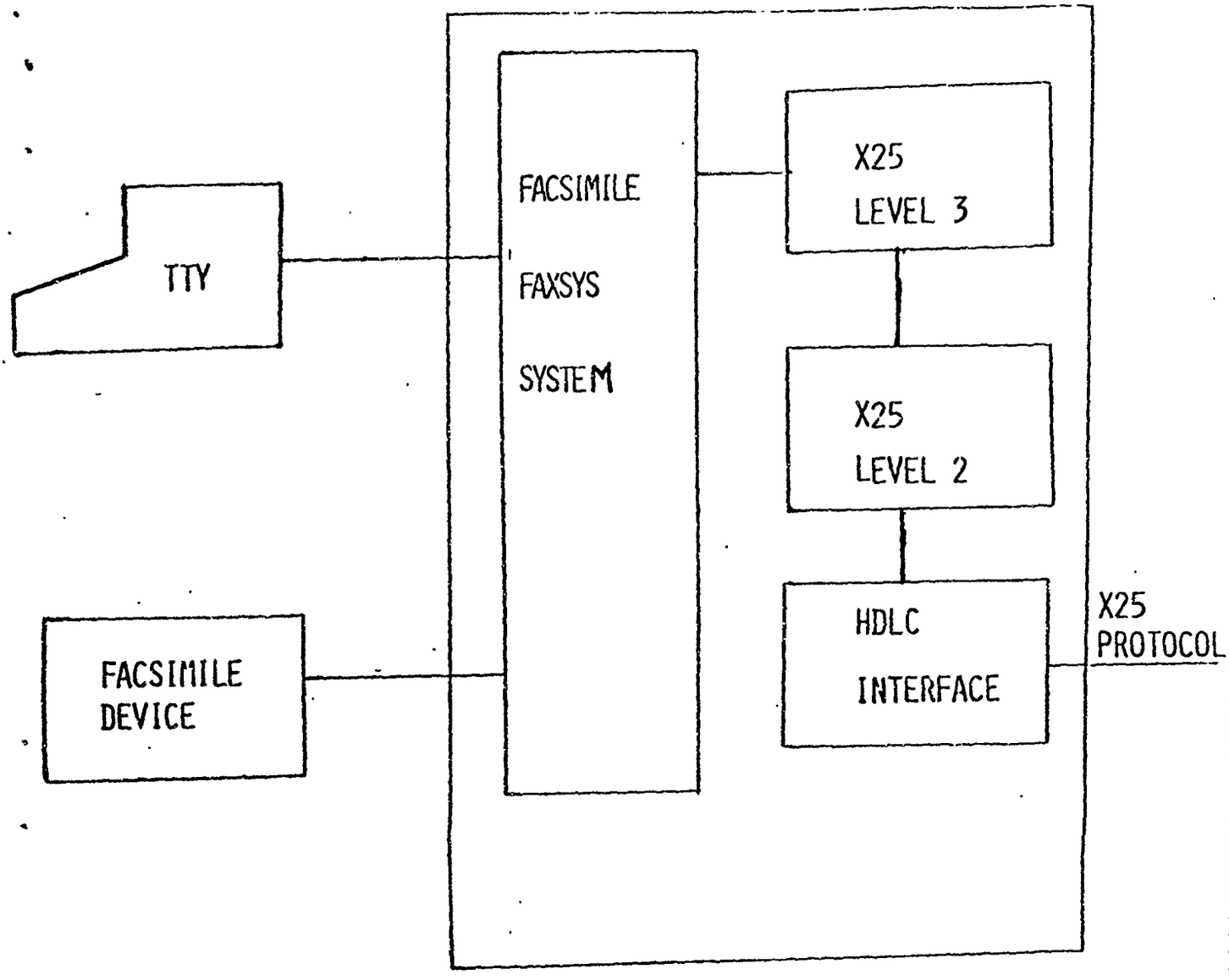


FIGURE 4.4: STRUCTURE OF THE INTEL 8080 SOFTWARE

4.4 Progress with the Experimental Systems

During this year, all the modules of the Experimental Test Host software of Fig 4.3, were made to operate successfully, except the HDLC interface level. The Level 2 and the X28 software were developed and tested also. All these have been written in a high level language RTL/2, and tested using a test harness on a PDP-11/34. A considerable body of documentation has been written. The sizes of the different modules (in K words) are as follows:

Level 2 (6K), Level 3 (8K), X28 (5K), VTP (8K)

Eventually. the ETH will be implemented on our LSI-11. A 28K word LSI-11 with floppy disk is being used. No HDLC hardware is yet available for the machine - at least in a reasonable time scale, so that we are constructing our own based on the COM 5025 chip. The HDLC hardware was largely completed by the end of the year. Until the LSI-11 and its hardware were complete, the HDLC interface software of the left box in Fig 4.3 could not be tested in any case; the hardware of this module should be completed early in 1978.

The traffic generation and monitoring module will be developed as required to perform different tests. Initial versions exist already as part of the test harness used on the 11/34; they include options to either output monitoring continuously or to write it into file storage. This option will allow us to output monitoring to the floppy disk to analyse on a larger machine, or as a separate task on the LSI-11.

In the Facsimile Test Host of Fig 4.4, hardware and Level 2 software have been completed, and the Level 3 and Frame Handler have been partially coded; these will be completed early in 1978. The modifications to the FAXSYS system to allow the X25 communication to be used have been studied but not yet started.

Considerable progress has been made with the EPSS Network converter represented by the right box of Fig 4.2. The new HDLC hardware has been commissioned. The Level 2 software has been coded and tested. The coding and testing of the Converter have begun. By the end of the period the Converter was being tested with the EPSS modules to check out the upper part of the final system. Early next year we expect to put the complete system together and to begin testing with the LSI-11 and the INTEL 8080.

V SATNET ACTIVITIES

The progress on the SATNET programme in its entirety is described in the annual reports by LINKABIT. Our own activities have been restricted to two areas: high level measurements across SATNET, and the provision of X25 interfaces to SATNET. The former activity has been described briefly in the previous annual report - complete discussion of the tools we have developed and some preliminary measurements are described in a paper which will be presented at EUROCOMP 78. This paper is included as the main body of this chapter. The work on X25 is still in its infancy, but was discussed briefly in Chapter 4.

PAPER 2

A HIGH-LEVEL NETWORK MEASUREMENT TOOL

S. W. Treadwell, A. J. Hinchley, C. J. Bennett

Department of Statistics and Computer Science
University College London
UK

Abstract

This paper describes the design of a measurement tool intended to relate high-level network performance to the internal network behaviour. There are a number of fundamental design choices that have to be made. Artificial and real traffic sources are compared in terms of experimental usefulness and validity. The advantages and disadvantages of collecting data at many points in the communications path are also discussed. The design of a high-level traffic generator which simulates the major types of network traffic is presented. Data is collected by timestamped packets, which can collect information from any point in a network which can recognise them. An interface is provided which allows control and data collection to be carried out in a distributed fashion, so that the experimenter may build controlling and collecting software appropriate to his own environment. The system has been implemented on the Atlantic Broadcast Satellite Network, and in principle can be implemented on any network requiring performance measurement software.

1. INTRODUCTION

The design of a computer network should include the provision of facilities for assessing its behaviour. These facilities can be provided in many different ways, depending on both the tradeoffs involved and on the ultimate aims of the measurements being conducted.

The measurement technique presented in this paper, called GNOME, was developed in an attempt to assess the performance a network can give to a real user. We wanted this tool to be able to explain this performance, not merely to observe it. The resulting design is a sophisticated system which is capable of individually observing the effects of many levels of network software on separate users.

In section 2 we discuss the fundamental design choices involved and the limitations of each approach. Section 3 discusses the actual design of the GNOME while section 4 discusses its implementation on the Atlantic broadcast satellite network (SATNET). Some conclusions are presented in section 5.

2. APPROACHES TO MEASUREMENT

2.1 Real vs Artificial Traffic

Since a network is essentially a communications medium, all measurement tools must be based on observations made on traffic passing through the network. This traffic may be from real users, performing real tasks, or it may be artificially generated traffic.

Monitoring of real traffic has the advantage that it is possible to observe directly the performance the network is delivering to its users. On the other hand, unless monitoring is performed at a very low level, it is difficult to use it to probe the internal behaviour of the network, since one cannot control the distribution of the traffic, and therefore cannot repeat a particular pattern at will.

This disadvantage is overcome by the use of artificial traffic, which by definition can be exactly and repeatedly described. For this reason, artificial traffic has almost always been used in experiments concerned with the modelling network behaviour and investigating flaws in network structure (e.g. Kleinrock76). However, artificial traffic suffers from the problem that performance figures gained by this technique require careful justification or validation before they can be assumed to reflect behaviour for real users. Since previous approaches have concentrated on solving problems connected with the mechanics of the networks, this question has not been explicitly addressed; the tool described in this paper is an attempt to find an answer to it.

2.2 Subnet vs End-to-end Traffic

Given a decision to build an artificial traffic generator, the next critical choice concerns its relationship to the network (or connected networks) being measured. In order to probe low level network behaviour more effectively, existing traffic generators have been built in close association with the low level network software. In the case of the ARPANET they are an integral part of the IMP (the ARPANET switching node). The advantages of this choice of measurement tool have been amply demonstrated by the work of Kleinrock quoted above. However, this approach does accentuate the validation problem discussed in section 2.1, as it does not emulate two significant factors which affect real traffic.

The first is that by the time real traffic has reached the subnet, it has gone through a significant amount of multiplexing, and will be demultiplexed on emerging at the destination. While a subnet generator can emulate the multiplexed traffic stream in the subnet, it is not clear that it can successfully emulate a single component of that stream. Secondly, real traffic must pass through several layers of application and transport protocols which are intrinsically invisible to the subnet. This makes it very difficult to emulate the effects these protocols have on traffic distribution. It also makes it impossible to use subnet generators for experiments with higher level protocols.

The alternative choice, investigated in this paper, is to build a traffic generator at the same level as sources of real traffic, i.e. in the host machine. This traffic will clearly undergo the same massaging by protocols as real traffic, and if properly designed it can simulate multiple activities. An approach of this kind has been adopted in the National Bureau of Standards measurement project (Abrams76), but this has treated the network as an entity to be measured purely from the outside, and so can only give global performance figures. It is unable to aid in pinpointing areas inside the network which are critically affecting the performance, although in fact the traffic generated does pass through all levels of protocol. In order to observe lower levels in practise, a certain amount of cooperation and recognition must be built into them. The extent to which we wish to do this is closely related to the type of data we wish to collect from the lower levels.

2.3 Statistics Collection

The next fundamental problem is the acquisition of data from the network. Two questions have to be decided - firstly, the nature of the data to be extracted, and secondly the means by which it is obtained. It is instructive to look at the solutions adopted by the ARPANET measurement tools to these problems. Data is returned to the user in two forms - cumulative statistics (CUMSTATS) and trace packets. CUMSTATS contain much detailed information on the traffic

situation as seen by a particular IMP - figures on buffer utilisation, number of retransmissions, internal queue lengths etc are all available. Trace packets, on the other hand, are effectively a picture of the situation seen by a packet in transit, giving information on the route that packets have followed.

These mechanisms, as they stand, are admirably suited for providing detailed information about the operation of a particular level of protocol (in this case, the IMP to IMP protocol). In the tool under consideration, our primary interest is in the performance provided by each level of protocol, and hence the main figures we are interested in are parameters such as throughput and delay. Building CUMSTATS-like facilities at every point of interest en route, or building interfaces to existing CUMSTATS, is a very major undertaking, and much of the information that could be obtained in this way is of very little relevance to any end-to-end effects. Generating trace packets also causes problems for a high-level tool, as trace packets may be generated by many levels of protocol, causing a flood of data to appear in the network which will make its apparent performance much worse than its real capabilities.

For these reasons we adopted a different approach to obtaining data. Our principal aim was to determine the effects of various components on the performance seen by the user, to determine (for instance) whether a given protocol implementation was a major bottleneck. We decided to insert TIMESTAMPS in the experimental packets which recorded the times at which these packets reached certain points in the route from end to end. From the differences between timestamps in the same packet one can determine the delays encountered en route; from the differences between timestamps inserted at the same point in successive packets one can determine the throughput figures. By selecting certain timestamps only, we can examine the behaviour of a specific level of protocol; by inspecting certain simulated users only, we can observe the behaviour of particular traffic patterns within a multiplexed stream.

There are some difficulties introduced by using timestamp packets. Lower levels must be modified to recognise requests for timestamps (although this is trivial compared to implementing a complete CUMSTAT interface), and each time a timestamp is inserted the packet checksum may have to be recalculated. There are also some limitations placed on the size of experimental packets. These must be large enough to contain all the timestamps that are expected. On the other hand, if packets are so large that they may be fragmented (which is particularly possible in internet experiments), timestamp information can only meaningfully refer to the first fragment. Finally, to obtain accurate one-way delay measurements, the clocks of all the various machines involved must be synchronised.

3. DESIGN OF THE MEASUREMENT SOFTWARE

From the decision to produce a high-level measurement tool, the design was attacked on four fronts. Firstly we had to determine exactly what would constitute high level traffic; secondly, we had to decide on how this traffic was to be entered into the network; thirdly, we had to develop a method of measuring the effect of traffic flow and finally we had to develop a technique for controlling experimental traffic.

3.1 Traffic Types

To model user network usage effectively it was necessary to determine the various types of user activity on the network. In particular we wished to classify the types of activity by the end-to-end exchanges required.

Three main traffic classifications are proposed. Variations within these classes, which the measurement software will accept, cover a wide spectrum of network activity.

3.1.1 Interactive Traffic

We classified the traffic typified by a user interacting with a remote process on a network as interactive traffic. Such a user activity might be accessing a data base or editing a file. In general these interactions consist of the user entering relatively small amounts of data, perhaps just a single line, and a short time later receiving a response from the remote process, even if it is only a prompt for the next input. In most cases the user will not enter the next input until some response has been received from the first.

3.1.2 Bulk Traffic

Bulk traffic involves the transmission of large amounts of data, such as a file, from one site to another on the network. Unlike interactive traffic, data only flows in one direction, from source to destination, and the data is usually sent as rapidly as the source destination and the network will allow.

Bulk traffic places a heavy burden on networks, mainly because of its high demand for buffers and line bandwidth, which can also have disruptive effects on other traffic. Disruption of other traffic often makes it desirable for the network to employ strict flow control for bulk traffic and perhaps to treat it with low priority.

Bulk traffic is not usually as time critical as interactive traffic. File transfers across a network for instance, may operate between two processes and not involve a user waiting for response as with interactive traffic.

3.1.3 Stream Traffic

The last type of traffic, termed stream traffic, is a one way transfer of data as in bulk traffic but the data is time critical and the volume of data relatively low. Telemetry data falls into this category, especially if real time control of some process is required. Another example is packet voice traffic where digitised voice packets are entered into the network at precise intervals and converted back to speech at the destination. Clearly, a late message cannot be "played out" and must be discarded. Volatility is the main characteristic of this type of traffic.

Stream traffic normally enters the network as fixed length messages at precise intervals. There is a valid lifetime for each message, after which the data is useless and can be discarded. (Ideally, if the network was able to detect such late messages en route, it should discard them immediately.) Messages can be delivered early to the destination but never late.

As the lifetime of messages is very short, the network often employs no flow control or retransmissions for this type of traffic as this would delay it too much. Thus stream traffic is very susceptible to interference, especially when bulk traffic is using the same data path. The amount of interference depends on the fairness control exercised by the network.

In terms of these traffic types, then, we can state that the purpose of the measurement tool is to see how well the network can handle each of these traffic types under both low load and heavy load conditions, and to study the amount of, and reasons for, interference between different traffic types.

3.2 Traffic Generation and Collection

To generate traffic according to the above descriptions and collect the statistics, we wrote a program that could perform both roles. This program, which we call GNOME (as the network we were using already had IMPs and ELFs), resides at a host level in the network. User activities to be simulated by the GNOME must conform to either interactive, bulk or stream traffic types. Generated traffic for such an activity, which we call a NETACT, is always sent to another GNOME which acts as the receiver. The response from the destination GNOME depends on the actual traffic type, and the details of the response expected from the GNOME are encoded on the artificial data messages.

Many GNOMEs may take part in an experiment, and each GNOME has the ability to simulate over thirty NETACTs simultaneously, with perhaps each one destined for a different GNOME. GNOMEs can play both the generation and receiving role at the same time.

Each NETACT is controlled by up to 17 parameters. The actual

parameters required and their use depends on the traffic types being simulated. The parameters control the frequency of message transmission, the message length and a host of other aspects of traffic generation. A list of the parameters is shown in Figure 1.

Parameter	Used by Types	Units
Traffic type	All	Coded Selector
Traffic destination	All	Coded Selector
Timestamp selector	All	Bit-coded
Delay before startup	All	Seconds
Duration of traffic generation	All	Seconds
Amount of traffic to be sent	All	Packets
Average delay between transmissions	All	Milliseconds
Allowed variation in delay	Interactive	Milliseconds
Average packet size	Stream and interactive	Bytes
Allowed variation in size	Stream and interactive	Bytes
Average delay before reply generated	Interactive	Milliseconds
Allowed variation in reply delay	Interactive	Milliseconds
Average size of reply packet	Interactive	Bytes
Allowed variation in reply size	Interactive	Bytes
Maximum outstanding data	Bulk	Packets
Maximum packet lifetime	Stream	Milliseconds
Packet priority	All	Coded Selector

Figure 1: Traffic Generation Parameters

3.3 Statistics Collection

Each GNOME has a statistics collection section that is independent of the traffic generation. Statistics are always collected on a per NETACT basis. The main form of statistics gathering is by the use of histograms.

Each NETACT in a generating GNOME is assigned a unique number, specified by the experimenter. This number, together with the GNOME number, is encoded in each message generated, enabling the receiving GNOME to identify a particular NETACT.

To collect statistics, the receiving GNOME is told the NETACT and GNOME number that statistics are requested for. A "statistics number" is also given that indicates which piece of information about the messages for that NETACT is to be histogrammed.

There are over 20 statistical values collected for each message received, and the histogram may select any one of these. Furthermore, there are various options permitting the experimenter to histogram the difference between two values, or the difference between the same statistic in successive messages which is particularly useful for analysing timestamping results.

This high degree of flexibility in histogramming is essential for the GNOMEs to be used effectively. The user is able to select any one of several different aspects to be inspected in detail: there is no limit to the number of histograms that may be collected for each NETACT. As the histograms are set up using a simple command, the statistics collection can easily be modified as the experimenter's attention moves from one statistic to another.

Normally, histograms are sent back to a collecting site at the end of each experiment. However, the experimenter need not be restricted by this. The histograms may be returned at the end of a specified interval. At the end of each interval, be it a few seconds or several minutes, the histogram contents are sent back and the histogram entries in the GNOME set back to zero ready for collection in the next interval. In this way the experimenter can not only get cumulative results over the entire experiment run but also results for each interval, allowing the variations in network response to be seen as the experiment proceeds.

3.4 Controlling the GNOMEs

For GNOMEs to be a useable experimental tool, an interface has to be provided between them and the experimenter. Several functions which must be included in this interface have been identified and separated into distant commands. These are:

i) SEIZE - Establishing communication with the GNOMEs.
ii) LOAD - Describing the traffic to be generated.
iii) LOG - Describing how the data is to be collected.
iv) START - Causing traffic generation to commence.
v) STOP - Causing it to cease.
vi) RELEASE - Terminating communication with the GNOMEs.

In order to provide an interface which is completely position independent, we assumed that the user may only be able to access a given GNOME from a remote host, which may even be in a remote network. For this reason, we decided that user/GNOME communication need only take place with one GNOME (designated the Head GNOME). When communication is established, the Head GNOME is provided with a list of all other GNOMEs participating in the experiment, and with the aid of this list it forwards all subsequent commands to the GNOMEs which are the ultimate recipients. Communication from the GNOMEs to the user is also routed through the Head GNOME. When it is satisfied that a command has been executed (or that it cannot be carried out), the original command is echoed to the user, with a reply field set to indicate what

has happened.

The commands in the above list were given appropriate packet formats, using 'raw internet' datagram headers to make transnet communication possible (Burchfiel76). The format for the SEIZE command is shown in Figure 2. In order to render this control structure directly accessible to a user, a controller program must be built which interfaces between user commands and command packets, as well as keeping an appropriate record of events and informing the user of progress reported by the GNOMEs. The controller can easily be built in association with the data collection software which must also be built to provide a complete distributed measurement system.

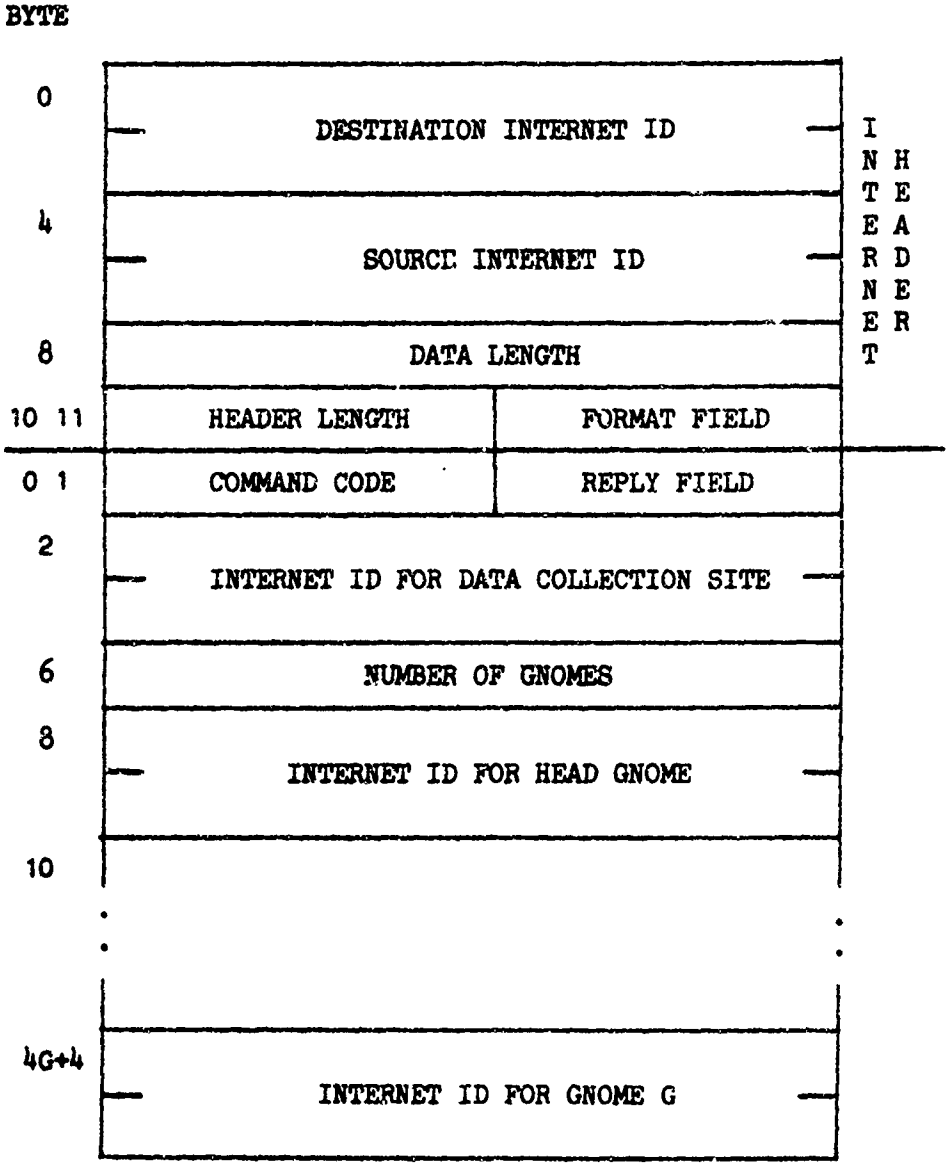


Figure 2: Format of SEIZE Command Packet

4. IMPLEMENTATION OF HIGH-LEVEL MEASUREMENTS ON SATNET

The current focus of the measurement system described here is the SATNET experimental broadcast packet-switched network, constructed primarily to investigate the design and measurement of network protocols appropriate to a broadcast satellite data channel (Binder75). This network, shown in figure 3, consists of four message processors (or SIMPs) at various sites communicating via an INTELSAT IV satellite. Throughout its history, SATNET has been closely associated with the ARPANET, and three of the SIMPs are connected to PDP11s which are in turn connected to the ARPANET as host machines (the fourth SIMP, a small station, is connected to a PDP11 which is in turn connected to the COMSAT IBM 370 at Clarksberg). These PDP11s are 'gateway' machines which translate between the host access protocols of ARPANET and SATNET, and forward transnet traffic from one net to the other. Currently there are no hosts on SATNET, and all traffic (except that from generators built into the SIMPs) must originate and terminate in the ARPANET. The primary functional role of SATNET is therefore as a transit net.

The measurement approach described here concentrates on using SATNET as a host network rather than a transit network. Gateways are conveniently located at the periphery of SATNET in a similar position to where one would expect to locate hosts. Moreover, gateways obey the network host protocol as it is at host-level that interconnection of networks is to take place. It is therefore appropriate to situate the GNOMEs as user processes on all the gateway machines in SATNET.

The fact that the gateways are intended to be used mainly in a transit role creates potential remote access problems; however, since they use a general purpose real-time operating system (ELF), the traffic generators were easily installed and tested above the gateway functions using a crossnet loader and debugger (Tomlinson76) which allows programs to be loaded, executed and debugged remotely from a TENEX host on ARPANET.

The SATNET GNOMEs are described in detail in (Treadwell77). We have taken advantage of the experimental nature of SATNET in two ways which fully explore the capabilities of the GNOME. Because most of the channel contention algorithms under study involve the use of fixed-length time slots, the times at which the SIMPs can transmit are already closely synchronised. This makes it easy to establish a synchronisation of the gateway clocks, and hence a global synchronisation for the whole of SATNET. It therefore becomes possible to coordinate the starting times of the various generation processes exactly and to extract accurate one-way delay figures.

Secondly, it is relatively easy to cause the SIMPs and the lower level software in the gateways to recognise timestamped packets, and to insert their own timestamps in them. In the GNOME design discussed above, timestamps can be inserted when the packets enter and leave

the GNOMEs. With this extra facility it becomes possible to test the ease and effectiveness of tracing a packet right through the network and examining its history in detail.

The controller we have built accepts commands from a user in ARPANET and relays them to the GNOMEs in SATNET, thus exercising the transnet control ideas discussed in section 3.4. Commands can either be entered in real time or in a batch mode by being taken from a previously prepared command file. In addition, the controller contains a command interface to the subnet measurement software, which allows us to change the channel contention algorithms, to set noise-levels for artificial packet corruption, to generate low-level artificial traffic, and to specify CUMSTATS to be returned. Coordinating the GNOMEs and the SIMP traffic generators in this fashion allows us to examine the importance of low-level effects for a high-level user, and to validate extrapolations to a high-level made from low-level experiments.

The same software also maintains a data file for the statistics generated by the GNOMEs, which are returned from all sites with access to the ARPANET during runtime. These statistics can be stored on disc or magnetic tape, or be dumped to the line printer for runtime inspection. Data stored on file is subjected to subsequent offline analysis on the IBM 360/195 at the Rutherford Laboratory. These routines are still under development, but typical output from an early version is shown in figure 4.

Since this controller/data collector was implemented on the UCL PDP9, it has the disadvantage that the data files cannot be made accessible through the network. A new version, which can be controlled remotely and also has a more sophisticated user interface, is currently being developed to overcome this problem. We at UCL are developing this controller under a TOPS 20 system on a PDP 20 at the Information Sciences Institute in Los Angeles, via ARPANET. The controller will be able to run on any TENEX or TOPS 20 system in the ARPANET, just as the GNOMEs can be run on any gateway PDP11.

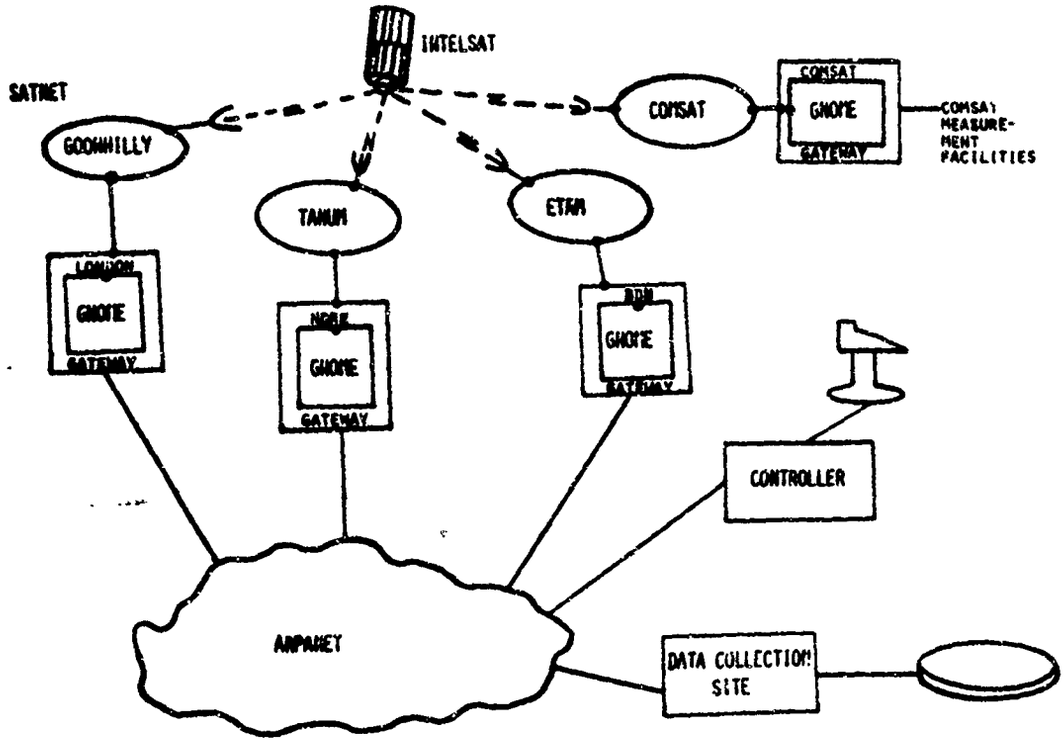


Figure 3: Schematic Diagram of SATNET

The diagram shows the position of the GNOMES in the various gateways, their relation to the four SIMPs (Goonhilly, Etam, Tanum and COMSAT), and the controller and data collection site in ARPANET. Points at which timestamps can be taken are indicated by dots.

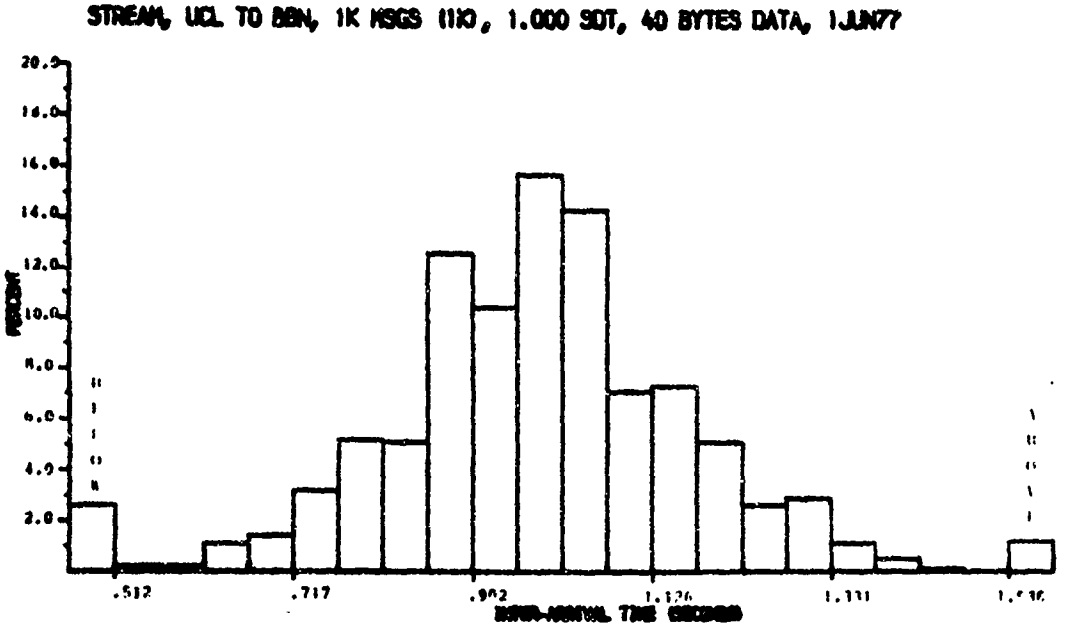


Figure 4: Sample Histogram Output

The figure shows the distribution of inter-arrival times for an experiment using stream traffic sent from UCL to BBN at 1 second intervals. As well as showing data within the observational range, the output also indicates how much data fell above and below it.

5. CONCLUSIONS

The traffic generator described in this paper is a high-level application oriented tool which can also inspect the effects of lower-level activity on higher levels. This aim is reflected in the design of the GNOME, most notably in its emphasis on simulating many network activities with diverse traffic patterns, and also in the generalised probing capability of timestamp packets.

A major concept behind the design was that the overall system should make as few assumptions as possible about the nature of the networks being examined and about the needs of the experimenter and the capabilities of his system. These requirements were met by placing the GNOMEs at a high level in the communications hierarchy and by presenting the user with a clearly defined interface to which he could attach control and data collection facilities to meet his local situation. By providing a clear separation from both the network and the experimenter in this fashion we have moved towards developing means for performance assessment which could be implemented on any distributed network, or even to examine connections which cross many networks. A standard technique for network assessment along these lines would be a major step towards developing 'benchmark' tests for networks of the kind used to commission mainframe machines.

The usability of the tool assumes a novel partnership between the evaluator of the network and its designer. While all traffic generation, data collection and control is external to the network, the timestamps must be put on by the nodes and the lower levels of protocol as the packets pass through. This type of facility is not being provided in any of the commercial data networks being developed by the carriers, though they are more in need than any other group of assessing their performance on both a global and detailed basis.

6. ACKNOWLEDGEMENTS

The cooperation of the workers of the Packet Satellite Working Group (PSPWG) is gratefully acknowledged. We would particularly like to thank Virginia Strazisar and Robert Weissler of Bolt, Beranek and Newman for their invaluable help with the SIMP and gateway software. This work has been supported by the Advanced Research Projects Agency under ONR grant N00014-77-G-0005 and the Ministry of Defence under grant AT-2047-064.

7. REFERENCES

(Abrams76) - M.D. Abrams et al., "Measurement of Computer Communication Networks", NBS Technical Note 908, July 1976

(Binder75) - R. Binder, "A Dynamic Packet Switching System for Satellite Broadcast Networks", Proc. Int. Conf. Comms., June 1975 pp 41.1 -41.5

(Burchfiel76) - J.D. Burchfiel et al., "Proposed Revisions to the TCP", INWG Protocol Note 44, September 1976

(Kleinrock76) - L. Kleinrock et al., "A Study of Line Overheads in the ARPANET", Comm. ACM, January 1976, pp 3 - 12

(Tomlinson76) - R. Tomlinson, "XNET, Cross-net Debugger for TENEX, User's Manual", BBN Report 3377, September 1976

(Treadwell77) - S.W. Treadwell et al., "GNOME User's Guide", UCL Technical Report 41, June 1977

VI MEASUREMENT ACTIVITIES

The measurement activity changed fundamentally during the course of 1977. Four measurement projects were terminated. The only ongoing one, with the SATNET has been discussed in Chapter 5.

No low level ARPANET measurements were carried out during 1977. However the previous work, which was mentioned in the last annual report, was finally written up (TREADWELL 1977). We would have liked to have made further measurements, to understand a number of performance problems we have been unable to explain. Unfortunately the ARPANET IMP-IMP formats changed very much late in 1976, so that the analysis programs would have required extensive modification. Since the person doing this work, Treadwell, was fully occupied with the SATNET measurements, we were forced to terminate this activity.

The usage measurements of the TIP from the PSTN were also largely abandoned. These measurements require dedicated use of a PDP 9; the pressure on our PDP 9s for other purposes became so severe, that no such measurement was possible after February. During February, a one week continuous measurement of all PSTN usage of the TIP was made. At the same time, the whole dialogue was captured of the use made of the NLM computer via the PSTN. Concurrently, the PO monitored the past usage during the same period. A report describing these measurements has been issued (STOKES 1977C). They showed, incidentally, the the PO measurements correlated well with our own.

The earlier 1976 measurements of both PSTN usage and of leased line usage via RL were analysed fully. The results are described in another report (STOKES 1977A). The usage via RL was continually measured whenever the link was functioning. These measurements continued throughout 1977. They were analysed also for the first half of 1977 in a third report (STOKES 1977B). Some typical results from that report are given in Figs. 8.1-8.3 of Chapter 8. This data is still being collected on a regular basis. We have been considering a more automatic method of transferring the data to RL for routine analysis.

A further project has been written up. For the previous annual report,(KIRSTEIN 1977A), our work on TCP measurement between UCL and Stanford U was described. Although the work was completed in 1976, it was finally written up only in 1977. The paper describing this work will be presented at the Conference on Network Protocols in Liege in February 1978. The paper provides the remainder of this chapter.

We have started obtaining usage and performance measurements on EPSS. This work is discussed briefly in Chapter 3. Little useful data has yet been obtained, because we have not started offering a really substantial EPSS service.

PAPER 5

MEASUREMENTS OF THE TRANSMISSION CONTROL PROTOCOL

Christopher J. Bennett
University College London
London, United Kingdom

Andrew J. Hinchley
University College London
London, United Kingdom

SUMMARY

The Transmission Control Protocol (TCP) is a protocol which has been proposed for process-to-process communication across connected computer networks. It sets up an association between two processes and manages the flow of data between them on a byte-based windowing principle. The first three implementations of TCP were made at University College London, Stanford University in California and Bolt, Beranek and Newman in Boston. Measurement tools based on timestamped packets and capable of running experiments in a variety of configurations were developed. Using these, the UCL group conducted a number of local experiments to examine the efficiency of our implementation, and some experiments across the ARPANET in conjunction with the Stanford group which examined the performance of the TCP in a real communication environment.

In this paper we describe the TCP, the experimental tools which were developed, and the configurations under which experiments were run. In practice, the experimental setup was not entirely adequate, and the reasons for this are discussed. The principal result of the local tests was that the UCL TCP was wasteful of processor cycles, and some of the experiments supporting this conclusion are presented. This was common to all three TCPs, and reasons for the inefficiency are discussed. Finally, the tests with Stanford revealed a damaging interaction between TCP and the ARPANET host-to-host protocol. The general applicability of this result to transnet communication is indicated.

1. INTRODUCTION

A critical service that must be provided in order to use a packet switched computer network is an end-to-end protocol for communication between two remote processes. Such a protocol provides standard mechanisms for performing basic communications functions such as the separation of various data streams, reliable sequenced message delivery, and the provision of end-to-end flow control. Typically, various services may be built into lower levels in the network, and those provided by the end-to-end protocol need only be a subset of the ones needed for end-to-end communication. In the ARPANET, for example, a great deal is done by the subnet between the source and destination switching nodes (known as IMPs), notably flow control and the generation and control of various acknowledgement and error conditions that can occur. The unique functions of the ARPANET NCP therefore are to manage the establishment of multiple connections, to respond appropriately to the information provided by the source or destination IMP such as error conditions and to provide host-host flow control.

For communication across connected networks, two main approaches can be taken. The first, which has usually been applied to virtual circuit networks, is to provide mappings between the end-to-end protocols of the various networks. These mappings are performed in the gateways which connect networks. The alternative approach is the 'Transport Station' approach, i.e. the provision of a universal end-to-end protocol, which generates and controls message flow in a standard transnet format. These packets are treated as data in each network, and are embedded in packets formatted according to the local network rules. In this approach, gateways support the transnet protocol by packet transport from one local net protocol to the next local net protocol.

It is essential in this approach that the end-to-end protocol chosen does not depend on the characteristics of the particular networks involved. The Transmission Control Protocol (or TCP: 4, 6) with which this paper deals, is one of the most detailed proposals made to answer these needs. Another example is the proposal known as INWG 96 (5). The first working TCPs were implemented at Stanford University in California, Bolt, Beranek and Newman (BBN) in Boston and University College London (UCL). A series of experiments and measurements were conducted between these sites.

The UCL work in these experiments concentrated mainly on three areas: the design of experiments and measurement software, the efficiency of the UCL implementation of TCP and the effectiveness of TCP as a transnet protocol. In section 2, we describe the TCP in outline and consider the experimental parameters and measurement tools in more detail. Section 3 discusses some efficiency measurements made locally, and some experiments conducted in conjunction with the Stanford group which are relevant to transnet communication. Section 4 discusses our findings in three areas of interest and section 5 summarises our main conclusions.

2. THE EXPERIMENTAL ENVIRONMENT

2.1 TCP

In this section we will briefly describe the version of TCP current during the period covered by these experiments. In TCP, communication is regarded as taking place between two processes which have unique socket names made up of a combination of net and TCP identifiers, and a port identifier, which establishes internal communication between the TCP and the process. These socket names can be reused for several connections to the source process, both to different processes at the same time, and to the same processes at different times. In this way it was hoped to avoid the complexities of the ARPANET socket allocation mechanism, although Tomlinson has shown (18) that problems then arise which lead to complexities in other areas of the protocol.

The two sockets set up an association using a sequence of packet exchange known as the three way handshake (7). As part of this, each side passes across a 'window size' and an Initial Sequence Number (ISN) which determines the 'left hand edge' of this window, and is the sequence number for the first byte of information to be sent out. The window size is an indication of how much data the receiving side is prepared to accept at any given moment, and may be varied at will to control flow. The left and right window edge (made up of the left edge plus the window size) determine the range of acceptable sequence numbers. Each byte of data and several control bits in the stream of information are assigned sequence numbers, starting consecutively from the ISN; as data is received it is acknowledged by indicating the next sequence number expected. This mechanism ensures that data can be delivered to the user without duplication and that sequencing can occur if it arrives out of order.

A process delivers data to the TCP as a whole or partial 'letter', where a letter is defined as a logical unit by the user process. The TCP then breaks these into 'segments', which are checksummed units of data encapsulated in an internet header, which contains protocol information such as the source and destination socket names, the current window size and sequence number, and various control bits. These internet segments are then treated as data by the various networks traversed. If they are too large for some network they are fragmented into smaller internet packets at the gateway on entry to that network. These internet fragments are similar to internet segments, except that they are not given an internet checksum, and the receiving TCP cannot deliver data to the user process until an entire internet segment is reassembled.

This description is intended merely to give the flavour of the TCP; the protocol is still evolving (6) and many details such as the desynchronisation and resynchronisation of connections have been omitted. These have often been introduced to enhance the security and reliability of the TCP where data or control is lost or delayed. Much of the theoretical work done on the TCP has consisted of proving that it is secure and reliable (e.g.17) in the face of the most unexpected sequences of events.

The UCL TCP was implemented on a PDP9 in Babbage, a high level assembler (16). The implementation was substantially derived from a BCPL version of the Stanford TCP, although it would have been very inefficient to implement this directly, due to the lack of index registers on the PDP9, and the run-to-completion nature of the operating system. Although it was the smallest of the first three TCPs (see Table 1), it was still a large program. However, subsequent implementations, such as the one at SRI (15) have proved to be significantly smaller and more efficient for reasons that will be discussed in section 4.

Computer	Size in K Words	Word Length in Bits
BBN PDP-10	9	36
Stanford PDP-11	11.9	16
UCL PDP-9	8.3	18
SRI LSI-11	1.9	16
UCL ARPANET NCP	4	18

Table 1 Comparative TCP Sizes (excluding buffers)

The figure for the SRI TCP excludes work space for handling connection information. The figure for the UCL ARPANET NCP only includes code handling protocol levels equivalent to the TCP.

2.2 Experimental Parameters

As the experiments were designed to test the effectiveness and constraints of TCP performance we were principally interested in examining the throughput achieved and the delays experienced by packets under various conditions. Very early in the program it was decided to isolate the TCP from the rest of the communications system; thus data was collected as seen by the TCP process, and the experimental parameters varied were ones which were specifically features of the TCP protocol. The parameters identified as being of particular interest were:

i Window size: This is the number of bytes of data the TCP is prepared to accept from a remote TCP. As was discussed in section 2.1, it is the basic method of flow control,

ii Maximum packet size: This parameter is determined more by the properties of the traffic and the transmission characteristics of the traffic medium than by an internal TCP constraint.

iii Thrash size: If data is being acknowledged at a very slow rate, the available window seen at the sender site may be very small. This leads to letters being segmented into many small packets. Thrash size is the parameter which defines the minimum size segment

the TCP will generate for transmission purposes.

iv Retransmission timeout: This is the length of time which passes before an unacknowledged packet is retransmitted. Ideally it should be set to just above the round trip time for the majority of packets.

A number of parameters are implementation dependent, or data dependent. These include items which are not easily quantifiable or variable, but which will have to be taken into consideration when understanding final results. The items considered in this category were:

i Packet size: As the TCP window is closely related to the total amount of information outstanding, it is important to study the effects on performance of the amount of data being transferred in any given packet. The limitations are those imposed by the need for a full timestamp field (18 bytes) at one end, (see section 2.4 below) and the maximum ARPANET packet size (90 bytes) at the other (a UCL implementation restriction).

ii Choice of acknowledgement strategy: It is possible either to force out separate acknowledgements (ACKs) for each TCP packet, or to piggyback ACKs on existing traffic in the other direction where possible. The choice of strategy may enhance or degrade performance, depending on the nature of the traffic flow; an ACK is needed to stimulate further transmissions, but may take up considerable bandwidth in a congested channel. UCL chose a strategy whereby a check was made that no packetised information was pending before forcing out an ACK. One may compare the two strategies by setting up appropriate traffic situations.

iii TCP buffering strategy: As noted above, the protocol places great importance on the relationship between window size and buffer constraints. As the measurements opted for a fixed window size, they also opted for a fixed buffer strategy and the emphasis was on determining a suitable relationship between the two.

The buffer strategy chosen, which proved flexible enough for most requirements, was to maintain a large fixed buffer pool from which fixed size blocks could be chosen as needed. Space was obtained according to demand for transmission, and for reception was chosen so as to maintain at least one outstanding receive buffer. If serious space limitation problems occurred, the experiment was abandoned; this tended to occur in the UCL experiments with larger window sizes as there was no necessary connection between the arbitrary window size chosen and the amount of space available.

Finally, in order to observe the interaction between TCP and the ARPA subnet, the packet type was varied. This is a ready-made control for examining the effects of subnet flow control on TCP performance and vice versa. Type 0 ARPANET packets are defined as data packets subject to the end-to-end flow control; type 3 data packets are those subject to none. However, type 3 packets run the risk of being discarded at any point en route if congestion is experienced.

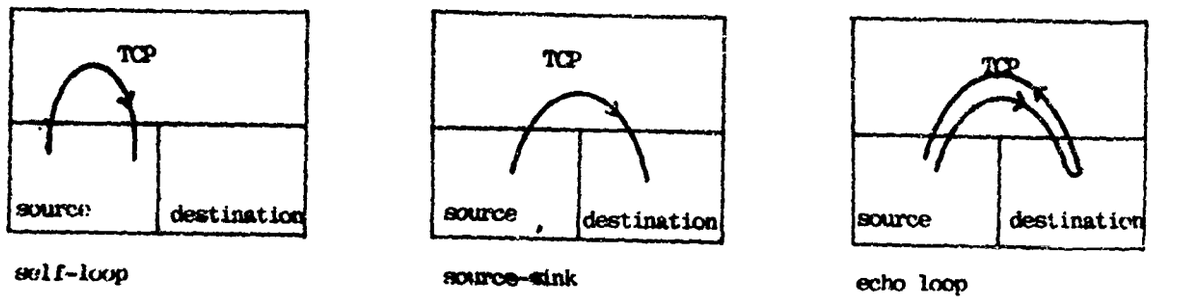
2.3 Configurations

Three basic traffic situations were identified which it was felt could give information on different aspects of TCP performance. These were: the self-loop, the source-sink and the echo-loop. In the self-loop, transmitted packets were received on the same TCP port. In the source-sink case, traffic was transmitted from one port and received on a different port which did not reply beyond acknowledging it. Finally, the echo-loop returned traffic to the sender.

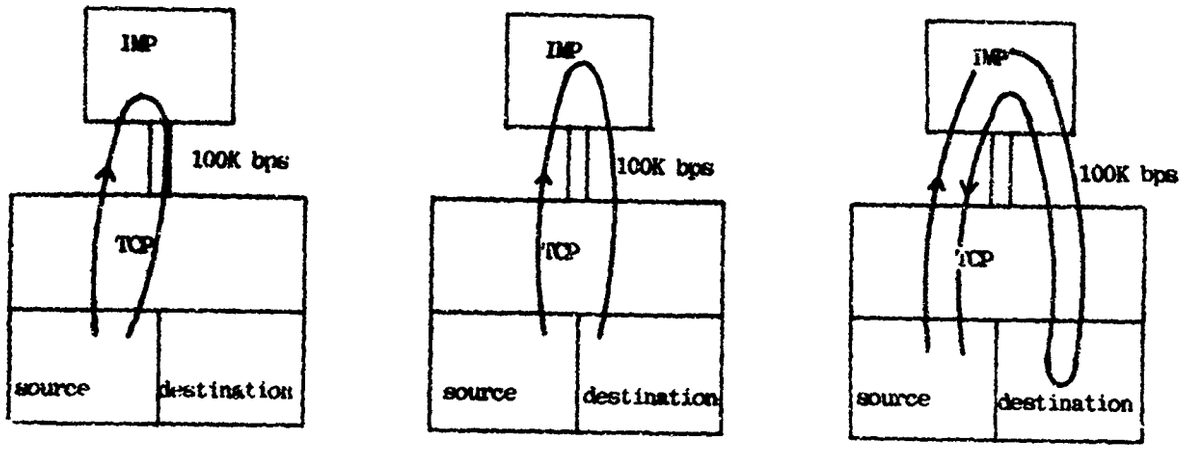
The self-loop was used to give an indication of the optimum TCP performance in various situations. The TCP is doing a minimal amount of work. As it is only transmitting and receiving on the same socket all acknowledgements are automatically piggybacked and no extensive computation is required. The source-sink and the echo-loop provided more normal situations, reflecting full and half duplex connections. The source-sink case by itself, however, could not give information on end-to-end delays to a remote TCP; this could only be extrapolated from round trip figures obtained by using an echo connection. In order to obtain usable data echo connections were almost always used for experiments to remote sites.

Behaviour was studied in several different physical configurations, illustrated in figure 1. Two local loops were used. In the "internal loop", TCP packets were placed on a transmission queue, and when they reached the head of it were immediately transferred to the receive queue. In the "IMP loop", packets were transmitted to the London IMP (switching node) across a local host interface. Traffic was also sent to a remote site, usually Stanford, although some connections were also made with BBN.

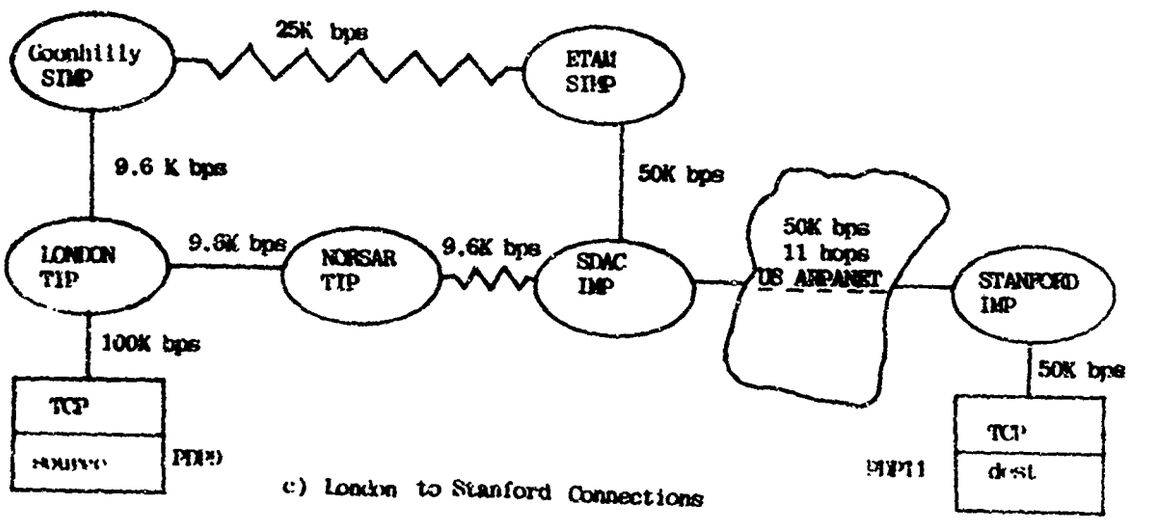
The UCL site is in an unusual position relative to the bulk of the ARPANET, in that it is connected by satellite links rather than 50K bps landlines. The standard link goes via NORSAR in Norway; another route used a broadcast satellite which is the vehicle for the SATNET broadcast satellite experiment (2). Both configurations were studied. The standard NORSAR route is a channel whose rated capacity is 9.6K bps, but which has an effective data capacity of between 5.1 and 7.6K bps for ARPANET host-to-host packets depending on whether seismic data is being transmitted from NORSAR. This channel is clearly a bottleneck, and one can expect delays to build up at NORSAR and SDAC on the return journey. The other configuration, using a 50K bps channel provided for the broadcast satellite experiments (when using the non-contention TDM algorithm), has a one-way capacity of 25K bps. The effective capacity of this link was only 19K bps, however, due to the effect of various constraints imposed by the communications hardware. In this case the potential bottleneck is the 9.6K bps London-Goonhilly link.



a) Internal loops



b) IMP loops



c) London to Stanford Connections

Figure 1. The Experimental TCP Configurations

2.4 Experimental Tools

To compute throughput and delay, one has to know the times at which various events occur. The critical points are the times at which a packet enters and leaves the TCP. Thus the basic tool for measuring TCP behaviour was a TIMESTAMP packet, which picked up clock values (or timestamps)at those points. As packets might be sent to either a sink or an echo process, the following times could be noted:

1. Generation at source process
2. Transmission from source TCP
3. Reception at destination TCP
4. Reception at destination sink/echoer
5. Departure from destination TCP
6. Transmission from destination TCP
7. Reception at source logging process

By including an offset field to point to the first unstamped location in the field, the timestamping procedure was made simple to code and operate. The format is illustrated in figure 2.

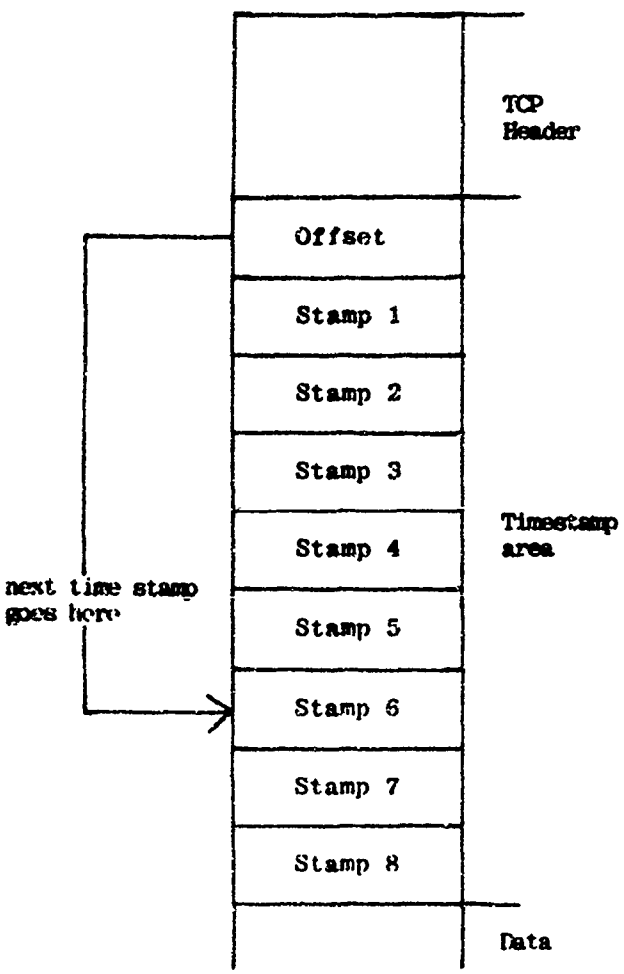


Figure 2 Timestamp Packet Format

Throughput can be calculated from the differences between corresponding timestamps in successive packets; delays from the difference between timestamps in the same packet. Throughput was measured directly in terms of the number of letters transmitted per second by the source process, where a "letter" was made equivalent to a packet for the experiments. The procedure suffered from a number of limitations.

a) The limit of clock resolution at UCL was only 50ms. This made some delays, such as round trip times in local configurations, practically invisible to the trace packet, and indirect methods were adopted to determine these.
b) The timestamps are purely TCP oriented, so effects of the environment - host operating systems, the network traffic etc. - are not easily observed without additional timestamps (which raise considerable implementation problems), or additional measurement tools. It was quickly found that in order to understand what was happening in the TCP in tests between UCL and remote sites, one would need additional details. To obtain these, two levels of packet tracing were used. One, adopted at both UCL and Stanford, was to dump each TCP packet on transmission and reception. This was used principally in debugging and in various robustness demonstrations beyond the scope of this paper, such as demonstrating the correct behaviour of various protocol features. In addition, packets passing through the London IMP could be traced using monitoring techniques developed for another project (19). Owing to the resources needed and a number of hardware problems, this procedure was only used once successfully, but it proved extremely useful in pointing out important and unsuspected TCP interactions.
c) The mechanism makes a meaningful study of TCP fragmentation difficult. Fragmentation of the timestamp field would cause loss and overwriting of timestamping information. Fragmentation in the data field means that only one packet can be traced, unless additional timestamp fields are created for each fragment and in that case interpretation of this additional information is unclear. Although there are meaningful experiments which can be performed in the presence of fragmentation, no attempt to do them was made in the measurements reported in this paper.

In order to support the facilities described above, UCL developed an integrated set of software to maintain the following TCP processes:

i) Manual exerciser: This process opened and closed connections and provided basic message transmission facilities. It was intended primarily to demonstrate the correct functioning of aspects of the TCP protocol.
ii) Echoer: A passive process was maintained which always had open a listening echo socket, and which echoed any data sent to it. It was intended to be used together with the remote parameter change facility described below.
iii) Traffic Generator: This process maintained a constantly blocked TCP transmission interface. Traffic of a fixed letter size was placed on the transmission queue whenever the TCP informed the process that the packet queue was becoming low. This mechanism ensured that traffic generation would occur at the optimum rate, which could thus be measured directly.
iv) Parameter Change: This was a special process which could request or implement parameter changes for both the local and remote

sockets of a connection, and thus in theory enabled a TCP experiment to be run by one side only. In practice it was only ever implemented at UCL, due to space restrictions at Stanford.

v) Data Logging: All incoming data was routed through this process which selected timestamped packets at regular intervals, monitored parameter change responses and maintained a watch over background information such as the opening and closing of a connection.

All these processes could be driven either from a command console or automatically from commands stored in a command file, written in a simple control language.

3. MEASUREMENT RESULTS

3.1 Local Tests: Implementation Efficiency

A large number of experiments were carried out in the two UCL configurations of figure 1 - internal loop and IMP loop. As a result of the early measurements undertaken, a number of serious inefficiencies were revealed, which were corrected, although the UCL TCP is still inefficient. It was found that in a heavy traffic situation the hardware data adaptor was idling for relatively long periods; although a simple rewriting of the TCP scheduler resulted in a dramatic improvement, the adaptor was still underutilised. The round trip figures of the UCL IMP loop represents an effective capacity of 26.5K bps across the local host interface, which has a nominal rating of 100K bps. This suggests that even in the comparatively idle situation represented here, the UCL TCP is compute-bound. Results in other experiments support this conclusion.

The results of a typical throughput experiment are shown in Figure 3. In this experiment, the self-loop was used to simulate a TCP running a single connection, and the echo loop to simulate two connection. Figure 3 shows the throughput seen by an individual connection in both cases. The experiment was repeated for internal and IMP loops. Adding a second connection to the internal loop reduced the throughput seen per connection by a factor of 1.86; for the IMP loop, the reduction factor was 1.85. The remarkable agreement of these two figures again supports a simple model of the UCL TCP, in which it very rapidly becomes process bound. In this model the throughput per connection for a number of connections each handling a similar load is inversely proportional to the number of connections. Such a model would predict the throughput ratio to be 2:1. The difference would be due to the minimum load of idle processing not being taken into account.

Throughput experiments involving variation of the letter size, and choice of the ACK strategy also suggest that in situations where the UCL TCP was being used for communications in a high capacity network, its use would incur considerable processing overheads. Although other groups had the same experience, this does not mean that the TCP is necessarily wasteful of processing cycles. The problem is largely due to inefficient implementation. Several areas where careful attention to efficiency is needed have been identified. They will be discussed further in section 4.2.

Configuration	Round Trip Time (sec)
UCL Internal Loop	0.073
UCL IMP Loop	0.107
Stanford via NORSAR	2.06
Stanford via Goonhilly	1.58

Letter size = 18 bytes, Thrash size = 18 bytes,
Retransmission time = 8 secs.
Window sizes = 18 bytes (to Stanford) and
= 256 bytes (local loops)

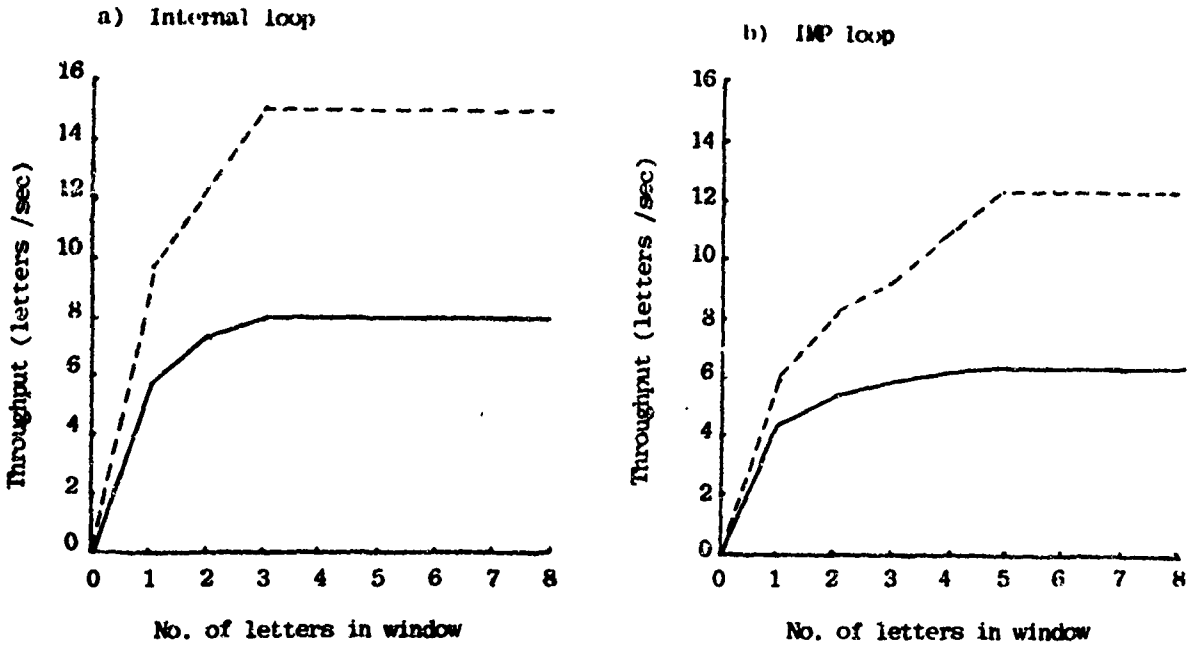
Table 2: Typical Round Trip Times for a Particular Experiment

3.2 Remote Tests: TCP as an Internetwork Protocol

A number of experiments were performed in conjunction with the TCP group at Stanford University which provided information on the TCP's ability to function in a more generalised network environment than was obtainable in the local UCL tests, (see Figure 1).

In this configuration, two factors were examined which are of interest in the transnet environment. The first was the fact that the communication medium used for the trans-Atlantic hop has substantially different characteristics from that on the US side. This situation is very likely to occur when two different networks are connected, and it is therefore useful to see how it affects the protocol's efficiency. Secondly, a TCP packet is considered to be entirely data within ARPANET, and must therefore be sent according to the normal ARPANET protocols. In practice the ARPANET link allocation procedure was bypassed and a special reliable datagram link was used. TCP packets were normally sent subject to full ARPANET flow control in addition to that of TCP. The alternative is to send packets which are not subjected to ARPANET control at all. As explained in section 2.1, the data packets that the ARPANET exercises flow control over are known as type 0 packets, and those with no flow control are known as type 3 packets. The use of type 3 packets is extremely risky, as there is a real danger of flooding the network with uncontrolled transmissions, and the only protection against this was the end-to-end flow control mechanisms of the TCP.

In these experiments, traffic was sent to Stanford under three sets of conditions. Type 0 traffic was sent across the NORSAR link, and type 0 and type 3 traffic were sent across Goonhilly. For all runs, minimum packets (18 data bytes) were sent



Letter size: 18 bytes Thrash size: 18 bytes
Retransmission: 8 seconds
Solid line: echo loop Broken line: self-loop

Figure 3. Throughput for One and Two Connections in Local Tests

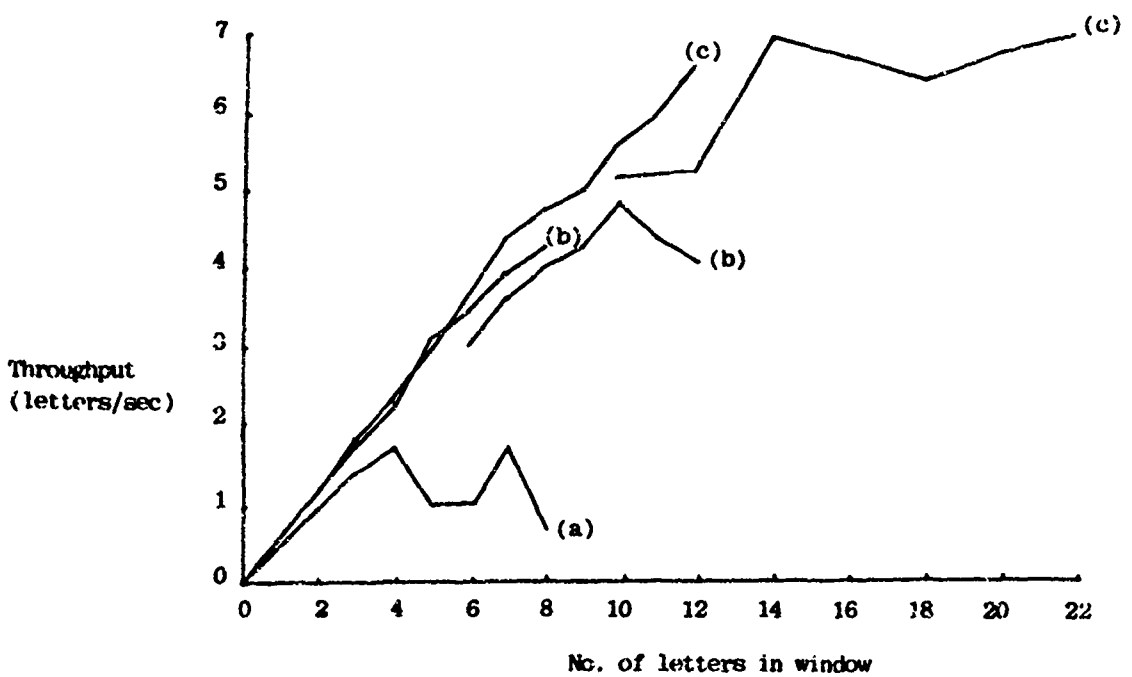


Figure 4. Throughput to Stanford

Letter size: 18 bytes Thrash size: 18 bytes
Retransmission: 8 seconds
Configuration:
(a) Type 0 via NORSAR
(b) Type 0 via Goonhilly
(c) Type 3 via Goonhilly

NB. Each line represents a separate experimental series

to an echoer at Stanford. The NORSAR run shown here was accompanied by the low-level packet trace mentioned in section 2.4. Table 3 shows figures on line level retransmissions so obtained and Figure 5 shows a sample packet history.

Minimum load round trip times are shown in table 2. A simple model constructed purely on the basis of known channel capacities (less overheads from headers, routing packets, RFNMs (ARPANET acknowledgements) etc) indicates that these delays contain no features of any significance at an internetworking level. The main constraint is simply the capacity of the channels en route. The throughput curves (shown in Figure 4) over Goonhilly support this picture. The maximum throughput to Stanford, 6.85 letters per second, represents a line-level throughput of 4.11K bps for letters of this size. The same throughput for full packets would mean a line-level rate of 7.9K bps, which would represent a near maximum utilisation of the 9.6K bps channel from London to Goonhilly. It appears that type 3 packets are thus limited only by the bandwidth of the channel. The effects of having two layers of "reliable transmission" protocol start appearing when we consider the throughput figures for type 0 over Goonhilly. Disregarding the rather anomalous peak observed at a window size of 10 letters, the maximum of 4.25 letters per second corresponds to a line-level rate of 3.26K bps and a predicted maximum of 5.7K bps, rather below the maximum channel rate. This would appear to be due to the maximum limit imposed by the ARPANET, of 8 IMP buffers for a connection, as throughput starts to level off at this point.

The full implications of the effects of having two "reliable transmission" protocols in operation become clear when we consider the information gained from the line-level management obtained on the NORSAR link. The picture here is more complex. The maximum throughput of 1.75 letters per second (1.34K bps at line-level) occurs at a window of 4 letters; this is followed by a severe degradation accompanied by a high level of IMP retransmissions from London to NORSAR. The effective capacity of the NORSAR-SDAC link is reduced considerably by the introduction of seismic data at NORSAR. A sufficiently large window in the source TCP will accordingly lead to an attempt to form a queue at NORSAR of more than 8 packets, the maximum allowed. This will cause an IMP-level retransmission after 2 seconds and a consequent degradation of throughput. IMP-level retransmission rates of up to 32.3% were observed (see table 3). TCP throughput is reduced further by the fact that features of the TCP are duplicated in the ARPANET protocols - in particular retransmission and positive acknowledgement. Since the IMP cannot distinguish a TCP retransmission as such, the London IMP can still be retransmitting a TCP retransmission of a packet after the UCL TCP already knows it has arrived successfully(see Figure 5). Thus in this sort of bad situation, the features built in to give end-to-end security lead to a duplication of low-level activity, increasing performance degradation as the duplication of packets can only be detected by the receiving TCP.

Table 3: IMP Level Retransmissions

No of TCP packets in window	No of TCP packets sent	No of IMP data packets sent	Percentage of IMP retransmission
1	-	-	-
2	127	129	1.55
3	-	-	-
4	117	118	0.85
5	126	181	30.39
6	149	220	32.27
7	127	144	11.81
8	-	-	-

Letter size = 18 bytes Thrash size = 18 bytes

Retransmission = 8 secs

Configuration: UCL to Stanford echoer via NORSAR

NB: This data refers to the points graphed in figure 4 for the NORSAR channel. The breakdown for data points 1, 3, and 8 is not available

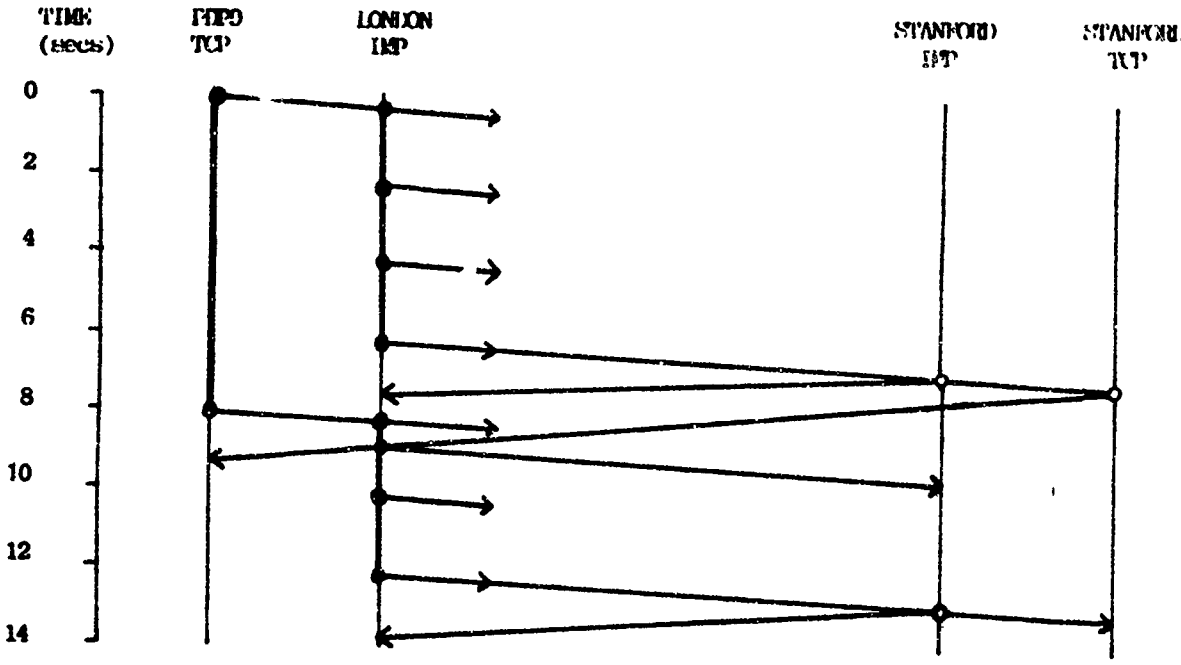


Figure 5. Sample Packet History Showing Effects of Protocol Duplication

The figure shows a simplified version of a packet history observed during an experimental run from UCL to the Stanford echoer via the NORSAR channel using type 0 ARPA packets. The dots indicate transmission of TCP data packets by the TCP or by the IMP. The circles indicate transmission of IMP or TCP acknowledgement packets. It is activity of this kind which accounts for the degraded throughput seen on NORSAR runs.

4. DISCUSSION

4.1 Approaches to Quantitative Measurement

In this paper we have outlined an attempt to measure quantitatively the effectiveness of a host-host communications protocol. To do this effectively both the performance of the particular protocol implementations one is dealing with, and also the characteristics of the communications subnet, must first be measured in a closed environment. The subnet is easier to measure, as the performance of a particular protocol implementation is a computer systems property which depends as much as it does with any other piece of software on a particular processor, operating system, and implementation approach. Measuring both elements together in the operation of the protocol across the subnet is a daunting task, which may explain why few measurements have been attempted of the kind described here. Measurement is however essential. Network suppliers may need to guarantee performance figures - Bell Canada have in fact done this for Datapac (9). Protocol design, although fundamentally driven by correctness of operation, can and must be influenced by efficiency considerations.

Nevertheless, making quantitative measurements in a real environment presents serious problems. Hamming is reported to have said (13) "without measurement it is impossible to have a science". In our experience, without vast amounts of time and effort it is impossible to make measurements. As mentioned in section 2.3 our experiments were performed either in self loops or in association with Stanford University. The latter path is one of the longest in ARPANET, being nearly 100,000 miles round trip, passing over satellite links and through approximately 11 switching nodes. Due to the time difference, experimental periods were limited to three hour stretches and the effort involved in coordinating activities and ensuring every component was functioning consumed a large proportion of available resources.

The exerciser developed at UCL was never used to its full potential. The remote parameter change facility and the file-driven experiment control were two unused features which in a more favourable environment would have permitted a much more rigorous series of experiments. Two machines available

locally at our site could then have performed many of the measurements unsuccessfully attempted with Stanford, without most of the organisational constraints. Robustness tests could also have been undertaken, involving the operation of the protocol for many hours to ensure that reliable sequenced delivery was maintained. Such tests were not possible with Stanford; in self loops they were not very meaningful.

The use of supportive tools was shown to be well worthwhile. The TCP packet trace was an essential debugging tool without which TCP would never have functioned. The value of the line monitoring tool is demonstrated elsewhere. A great deal of work was also done using simulation techniques to study areas of flow control which could not be covered experimentally (8). Much of the work reported in this paper could not have been achieved without the use of tracing techniques to give a qualitative explanation for the figures obtained from the main, quantitative, exerciser.

4.2 The Efficiency of TCP Implementations

The inefficiency of the UCL TCP reinforces the fact that attention to efficient implementation is at least as essential with communication drivers as it is with any other part of a system which is going to be used very frequently. A study of the overhead of supporting the normal host protocol (NCP) for ARPANET on Tenex machines indicated that the overhead was only 20% greater than that for the same traffic to local devices (14). Ideally, we should be able to limit TCP overheads to this sort of figure. The decreasing cost of CPU cycles will eventually ameliorate this problem, but it cannot resolve it entirely.

We can analyse the costs of implementing TCP by examining the various computational areas associated with supporting the specification.

i) Buffer Manipulation
To make the best use of our available space we must cater for varying message lengths by using dynamic free pools of varying lengths, and the resultant frequent garbage collection. Several free pools must be accessed using the standard system calls (with attendant overheads) for every buffer request, transfer, and return to free space. This overhead has been found to be heavy for TCP and BBN has proposed (3) that fixed buffer sizes should be used for any particular association, together with reassembly direct into the user buffer. This approach removes the need for dynamic pools of varying length, and also reduces the number of transfers required.

ii) Table Searches
Here we refer particularly to table searches associated with multiplexing connections. As TCP allows an association to be any unique source-destination address combination, no short address or index conventions are adopted for it. It is possible then to be consuming cycles in chaining down the connection list for each incoming message. Some intelligent hashing of addresses will reduce this overhead.

iii) Arithmetical Computations
TCP employs a large sequence number space to avoid the possibility of two letters in transit ever having the same sequence number. This requires 32 bit arithmetic to be performed both in generating a new sequence number, and in testing that an incoming letter lies within the current window. Checksumming requires the same computational support.

iv) The Place of TCP in the Operating System
We must decide how much of the protocol should reside within the machine's basic executive operating system, and how much should reside as a normal process. In many operating systems this distinction may make a considerable difference. Because the protocol may be supporting transfer of a high volume of information from the machine to the outside world and vice versa, we may prefer it to be in the executive space. However, because of its size, its particular use of buffer pools and its relations with user processes we may prefer to implement much of it as a user process, which will probably incur substantial overheads when performing operating system calls.

v) Choice of Language
Although the desire for a clear implementation makes the use of a high level language attractive for implementing TCP, known overheads of systems implementation languages are around 30-50% and can be much higher. Such overheads can be quite bearable for many applications. For a communications driver such as TCP, performing many actions per message, and even a number of actions per character, the cumulative effect is significant.

Overall, implementation experience suggests that all these factors must be taken into account. A recent LSI11 version of TCP has achieved very acceptable throughput figures (15). This version is written in assembler and uses circular buffers in a system with a minimal operating system environment.

4.3 Layered Architecture and Transnet Protocols

The design of the TCP is one of a class of end-to-end protocol designs which is based on a model of several distinct layers of protocol each performing certain clearly designed functions. The benefits of this approach have been pointed out in several earlier papers, both on TCP and the Cyclades/EIN transport station (5). The main advantage lies in the simple network structure produced when all responsibility for acknowledgement sequencing, reliable delivery, flow control and other features are concentrated at one level. This produces demonstrable benefits in line overhead reduction (12). In order for layering to be applied successfully, unnecessary duplication of protocol features in more than one layer must be avoided. In the design of a single network, this may be possible if the protocol designer can assume lower level functions as given. When a protocol is designed for transnet use, however, this assumption cannot be made. Duplication of function will almost inevitably occur, leading to

mutual interference of the kind we have observed.

This is clearly a general problem. For instance, transport stations communicating across X25 interface are being built using the INWG96 protocol (10) which is similar to TCP, and this effort could well show similar phenomena. It is clear that status and congestion information must be exchanged between levels of protocol. It is less clear how to do this, and what to do with the information. We may need to define a network-to-gateway interface which supplies this information, and a standard gateway-to-gateway protocol to allow for its propagation to source (1). Networks may even be required to provide a raw datagram interface to gateways on top of which a standard gateway-to-gateway could be imposed as necessary. This is clearly not always possible - the current version of X25 could not be fitted into this framework.

5. CONCLUSIONS

There are three major conclusions to be drawn from the work reported in this paper. The difficulties experienced both in making measurements and in co-ordinating the activities of the measurement groups highlight the importance of building satisfactory remote control facilities into the measurement software in this kind of project. It is clear that without these facilities the effort involved in undertaking experiments is out of all proportion to the results obtained.

Each of the three implementations involved in these measurements was notably inefficient. The processor-dependent reasons for this have been discussed. The weakness of the original TCP window mechanism was a major factor in this inefficiency, and the lack of suitable algorithm to control window size was noted by all groups. This has led us to initiate a set of simulations (8) to investigate the effects of various flow control strategies on TCP performance.

Finally the crude retransmission conflicts observed point out the kind of difficulties that can arise in a transnet situation if there is no strategy for interacting with lower level controls. It is clear that such strategies must be evolved and that these will probably be managed by the gateways, but at present discussion is very open on exactly what is needed.

ACKNOWLEDGEMENTS

The support of the Science Research Council under grant B/RG/67022 and the ARPA IPTO office under ONR grant N 0014-74-C-0280 is gratefully acknowledged. Additional support has been received from the Ministry of Defence under agreement A7/2047/064 and the NATO travel fund.

Much of the implementation work described here was done by Frank Deignan. The Stanford group: Yogen Dalal, Judy Estrin, Dick Karp and Jim Mathis were essential collaborators in the measurment process, and Bill Plummer was mainly responsible for the BBN implementation. Paal Spilling of NDRE was with us for part of the measurment period, and assisted with definition of some of the experiments. Finally, Professor Vint Cerf, the head of the Stanford group at that time, and co-designer of the protocol, gave us invaluable encouragement.

REFERENCES

(1) C. Bennett, S. Edge and V. Hinchley - Issues in the information of Datagram Networks. INDRA Note 637, Department of Statistics and Computer Science, UCL, July 1977.

(2) R. Binder - A Dynamic Packet Switching System for Satellite Broadcast Channels. Proceedings of the International Conference on Communications, p 41.1 San Francisco, June 1975.

(3) J. Burchfiel, W. Plummer and R. Tomlinson - Proposed Revisions to the TCP. INWG Protocol Note 44, September 1976.

(4) V. Cerf, Y. Dalal, C. Sunshine - Specification of Internetwork Transmission Control Program. INWG Protocol Note 72, December 1974.

(5) V. Cerf, A. McKenzie, R. Scantlebury, H. Zimmerman - Proposal for an International End-to-End Protocol. ACM Computer Communications Review Vol.6 p.63. January 1976.

(6) V. Cerf - Specification of Internetwork Transmission Control Program Version 2. September 1977.

(7) Y. Dalal - Establishing a Connection. INWG Protocol Note 14, March 1975.

(8) S. Edge - TCP Performance Measurements from Simulation. INDRA Note 619, May 1977.

(9) S. Erskine - Datapac: A Packet Switched Network for Canada. International Conference on Data Communications Networks, London May 1977 (to be published).

(10) Y. Jacquemart & A. Danthine - Loosely Coupled Interface between Transport Station and X25. The Third European Network User Workshop, IIASA, April 1977.

(11) R. Karp - TCP Experiment Plan. Private Communication, September 1975.

(12) L. Kleinrock, W. Naylor and H. Opderbeck - A Study of Line Overheads in the ARPANET, CACM, Vol.19 p3, January 1976.

(13) L. Kleinrock - Queueing Systems Vol.2. John Wiley 1976.

(14) J. Postel - Survey of Network Control Programs in the ARPA Computer Network. Mitre Corp. Report MTR 6722, June 1974.

(15) Packet Radio Net Development - SRI International Project Quarterly Report. TR 2325 May 1977.

(16) A. Stokes - A User's Guide to Babbage. UCL TR Report No.13, 1974.

(17) C. Sunshine - Interprocess Communication Protocols for Computer Networks. SU-DSL TR Report 105, December 1975.

(18) R. Tomlinson - Selecting Sequence Numbers. INWG Protocol Note 2, August 1974.

(19) S. Treadwell - The Measured Characteristics of Traffic in the UCL Node in the ARPA Network. UCL TR Report 33, December 1977.

VII FACSIMILE ACTIVITIES

7.1 Overview

The UCL Facsimile work is another activity which has reached fruition during the last year. The system of analog Facsimile device, use of Message Systems and the Data Computer have been completed. A paper describing this system is given in Section 7.2. The paper will be presented at EUROCOMP in London in May 1978. We have analysed the components of this system, and concluded what technological improvements are required to make such a system economically viable. This work is described in Section 7.3.. This is a paper which will be presented at the International Computer Communications Conference in Kyoto in September 1978.

A key aspect of the economic viability of Facsimile as a method of inter-person communications is the cost of the communications transmission itself. We have therefore analysed the tariff structure of several of the PTT supplied data and telephone networks. We conclude that their present levels are rather unattractive for Facsimile. This work (KIRSTEIN 1978) will appear in Computer Networks.

Although these aspects of the Facsimile work are completed, the project is entering a new phase. Hardware has been developed to allow the terminal to be attached to X25 networks; the software for this purpose is practically completed. This work, together with the X25 network developments of Chapter 4, should allow the Facsimile terminal to be attached directly to an X25 network - including these versions of EPSS and SATNET.

Other future aspects of the work being considered are modifications, hopefully being done by ISI, for incorporating some of the UCL features directly into forms of the TENEX message system and into the PDP 11s driving other XGP printers. We at UCL will also replace our analog Facsimile device by a digital one. We then hope to develop a very useful and replicable self-contained terminal.

PAPER 1

UCL EXPERIMENTS IN FACSIMILE TRANSMISSION

USING DATA BASE MANAGEMENT FACILITIES ON ARPANET

S. Yilmaz and P.T. Kirstein

Department of Statistics and Computer Science
UNIVERSITY COLLEGE LONDON

ABSTRACT:

The methods of message processing developed over ARPANET have been extended to handle facsimile information. The message is divided into two portions. Using archival storage techniques the facsimile portion is shipped to a large store at the Computer Corporation of America [CCA]. At the same time the corresponding text is sent to a list of addressees through the standard message service [MSG]. The cross- reference between the two parts is secured by means of an automatically generated header information, also describing the nature of the facsimile data for subsequent reproduction.

The prototype system at University College London [UCL] consists of a standard 4-6 minute analogue tranceiver front-ended by an Intel 8080 micro processor. The system uses the ARPANET Packet Switched Network facilities, and at present, is set up to communicate with a Matrix Printer [XGP] at the Information Sciences Institute [ISI] in Los Angeles. Since the facsimile data is transformed into a completely machine independent form, other facsimile devices could also be included with little difficulty.

This paper describes the system, gives examples of its use, and draws conclusions for the future applications of facsimile techniques in computer networks.

1 Introduction

The present message and text manipulation services offered over ARPANET are extremely important for processing textual data input directly into one of the Hosts on the network. However, these services have one element missing; there are no facilities for dealing with material which does not originate in a machine readable form, and the facilities for processing non-textual material are poor.

We are investigating the computer input of facsimile documents, their transmission, their storage and subsequent retrieval, their manipulation in conjunction with other files, and their output on devices different from those on which they were input.

Our work in these areas led to the realisation of a Facsimile Terminal Model which can be used to transmit facsimile as well as textual information within the ARPANET environment.

Our conceptual thinking about the integration of facsimile equipment into ARPANET, and the way we attempted to realise it, are discussed in Section 2. An essential part of our development was to put considerable intelligence into the facsimile terminal. This was accomplished by putting a Micro-processor between the standard facsimile device and the data network. Our short term progress and long term plans for this terminal and its uses are discussed in Section 3. A key ingredient of our concept is the use of an Information Storage and Retrieval node [IR]. For this purpose we are using the Data-Computer of the Computer Corporation of America. In Section 4. we describe the operation and the use of this machine as it relates to our requirements. In order to put our ideas into practice, specific experiments were undertaken involving ISI and ourselves only. There is a summary of these experiments in Section 5. Finally conclusions are drawn in Section 6.

2 The System Overview

In principle the system we have developed is shown in Fig. 1. The UCL facsimile terminal FAX1 is connected, via the TIP and a character terminal interface, to ARPANET [Ref. 1]. Also attached are: an information storage and retrieval system IR, a message processing system MP and another facsimile terminal FAX2. MP and IR in Fig. 1 are shown in different places, but there is no reason why they could not be accommodated within the same host. Our aim is to develop a system where the facsimile terminal is sufficiently intelligent that it can read documents with a text header and address list, to transmit and store them in IR, while notifying the addressees through MP that the document is stored in IR. Later the addressees may access IR and retrieve the file on their own facsimile device FAX2. At the recipients' option (or possibly the sender's to mimic "recorded delivery") each addressee may automatically inform the originator that the document was received. In addition, we wish to store the facsimile data in a canonical form, so that it may be retrieved from different facsimile devices.

There is an important difference between the storage of facsimile and short text messages. Such textual information is usually small (less then 500 bytes), whereas the facsimile data is much larger typically 25 K bytes for an A4 size page with optimum data compression. Hence the textual data can be stored in multiple copies, one for each addressee in his "mailbox". The facsimile information is stored only once in IR, and the path name to retrieve it is known by the addressees from their text messages.

In our implementation of this system we have used ARPANET as the data network. For FAX1, we attached a Micro-Processor to an analogue 4-6 minute facsimile device (Section 3). This was then connected via an asynchronous link to the UCL TIP. However, for reasons of speed and flow-control, this link was switched over to one of the UCL PDP-9 as described in Section 3.2.5.

For the MP of Fig. 1, we use one of the Tenex systems on ARPANET ,supporting the message system MSG [Ref 2]. For IR the Data-Computer at CCA [Ref. 3] with an on-line storage capacity of 10**12 bits, is a natural candidate, and has been used exclusively. As for FAX2 we are using the XGP printer [Ref. 4] at ISI, which is controlled by a PDP-11 Tenex configuration as described in Section 5.

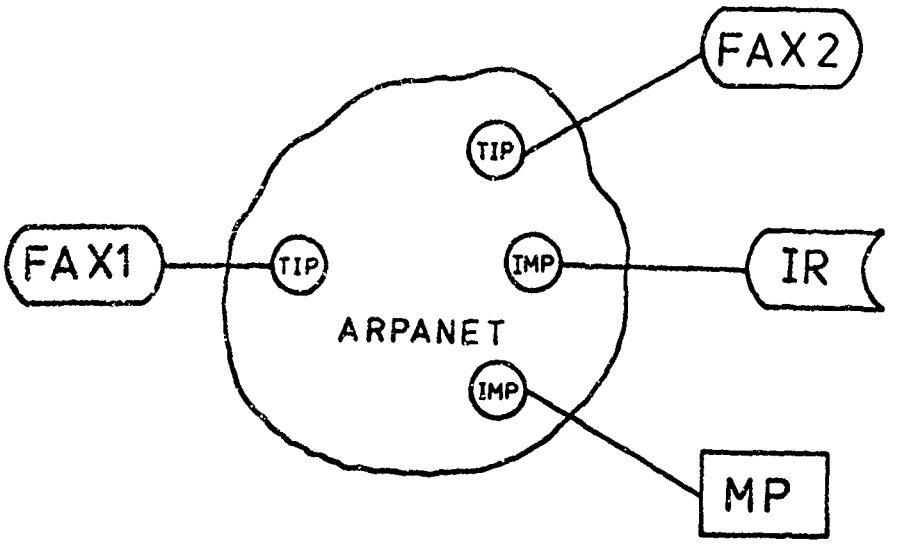


Fig.1 Overview of an Idealised Facsimile System

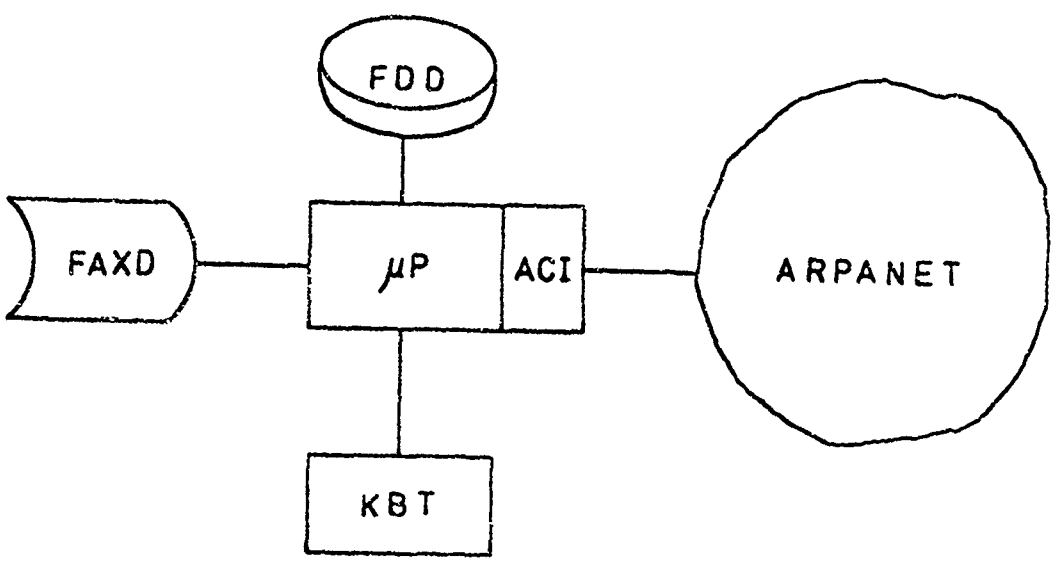


Fig. 2 UCL Facsimile Terminal

3 UCL Facsimile Terminal and its Use

3.1 Introduction

A system such as the one outlined in Section 2 can be described at many different levels. In this section we will describe the UCL facsimile terminal itself [Section 3.2] and give examples of its use in message transmission. The examples given contain two user dialogues, one for message transmission [Section 3.4.1] and the other for message retrieval [Section 3.4.2].

The message transmission dialogue is mainly concerned with the composition of a message file [containing textual and facsimile information], its local verification, its submission to the MSG system and the Data-Computer, and confirmation of delivery. This dialogue also illustrates the binding of linkages between the two portions of the message and the supervision of message transmission which are almost transparent to the user.

The retrieval of a message can have either one or two stages. Because the notification comes via the text message system MSG, the user may receive notification of facsimile messages during straightforward text retrieval from a conventional terminal. In this case the retrieval of the facsimile portion must be a separate exercise. Alternatively, as illustrated in the retrieval dialogue, the user may access the MSG system via the facsimile terminal. In this, somewhat simpler case, the notification and retrieval become an integrated operation.

3.2 The UCL Facsimile Terminal

Fig. 2 shows the basic components of the facsimile terminal we have developed. Physically it consists of :

[1] FAXD : The facsimile device which is a Plessey 4-6 minute analogue tranceiver [KD-111] incorporated with a two level Analogue-to-digital converter.

[2] uP : An Intel 8080 mico-processor with 24K [8-bit word] Random Access Memory, and peripheral interfaces.

[3] FDD : A Floppy-Disc driver.

[4] KBT : A Keyboard Terminal.

[5] ACI : An asynchronous communications interface.

3.2.1 Facsimile Device [FAXD]:

The KD-111 machine was modified for computer interfacing. The analogue signal from its scanner is tapped and "Amplitude Quantised" via a simple threshold detector yielding a two-level [Black & White] signal. The output of the detector is then "Time Quantised" at the rate of 2.4Kbps. This sampling rate gives a resolution of 96 pels/inch, equivalent to the fixed vertical resolution of the device itself. The resulting bit stream then has a start and stop bits added to each 8 data bits, and is fed to the Micro-Processor. Formatting the data in this fashion allowed us to use the readily available UART communications interfaces between the facsimile device and the Micro-Processor. A synchronous data adaptor will eventually be installed to replace the present asynchronous one. This is more appropriate for the usual situation where the terminal is remote from the network and must use medium speed data transmission facilities.

3.2.2 Floppy-Disc

Present analogue facsimile devices must synchronise the operation of the scanner and the recording mechanisms. In the case of the KD-111 this takes up to 15 seconds to accomplish, once the document scanning [recording] starts there can be no interruption whatsoever. In a packet switched network like ARPANET, it is impossible to guarantee any specific data throughput at any time. Therefore, to overcome these unpredictable flow conditions, facsimile data must be staged. We do this by using floppy-discs which can buffer up to 4-5 pages of facsimile information.

We are planning to replace the KD-111 with a digital facsimile device, in which case the data flow can be controlled in accordance with the network throughput conditions. Therefore, with the arrival of a digital machine, a floppy-disc should no longer be necessary, unless the provision of a temporary backing store becomes a desirable feature of the system.

3.2.3 Micro-Processor

The micro-processor of Fig. 2 is the brain of the facsimile terminal. One of the principal aims of our work has been to transform a passive facsimile device into an intelligent terminal. In the context of this work the word intelligence has a double significance. On one hand

intelligence is required to control the behaviour of a facsimile machine so that automatic starting and stopping of the device, re-scanning lines [even pages], threshold settings etc, can be realised. On the other, intelligence is required to free the user from lengthy routine operations that are inevitable when such devices are to be used in computer networks. Although we have covered much ground in achieving the latter objective, we have not been able to do the same with regard to controlling the facsimile machine itself. The reason for this is the finely tuned nature of present day facsimile equipment which are optimally designed to meet straightforward commercial needs. However, this situation is expected to improve with the release of the new generation of facsimile devices.

The micro-processor provides also the digital processing power needed for data compression and transformation for machine independent data representation as described in Sections 5.

3.2.4 The User Console [KBD]

The keyboard is needed to control the system. It provides a medium for the user to compose his text message , specify addressing information, and initiate transmission and retrieval of messages. In our system a standard teleprinter is used, but a simple keyboard would be quite adequate.

3.2.5 Network Interface [ACI]

Originally this system was planned for connection to the Terminal Interface Processor [TIP] ports, which have an asynchronous format. Accessing the Data Computer requires two separate links, one for control [Data Language], and the other for binary data transfers. We found, however, that due to poor flow control procedures , it was impossible to pass binary data through the TIP at any reasonable speed; under certain circumstances the TIP would actually lose characters. To overcome this problem we decided to attach the system to one of the UCL PDP-9s. This link was implemented on a single asynchronous channel accoutred with a simple "Binary Transparent" protocol, multiplexing several logical connections between the micro-processor and the network. This modification does not affect the basic architecture of our system in any way, it merely takes advantage of the Host-to-IMP interface of ARPANET for reliable transmission of facsimile information.

3.3 The User's View of the System

The user FAXSYS dialogues take place via the system console. A set of commands permit the user to interact with the user interface process. Input of a command causes this process to activate the associated segments to perform the required operation. Our earlier work on a similar implementation [Ref. 5] had shown the importance of shielding the user from any fragmentation , and providing him with an understandable and unified view of the system. Therefore, the user is informed about the progress of his requests, but he is in no way made conscious of the complex operations behind the scenes.

Table 1 at the end of this paper shows a list of the FAXSYS commands currently available, their meanings, and the corresponding working parameters. Each command is formed by a "#" sign followed by the first letter of the most significant word in the command phrase. The user can input more than one command for simultaneous operations. For instance:

```
+----------------------+
!  #L ISI, #E <CR>     !
+----------------------+
```

command string would log the user to the MSG system at the Information Sciences Institute in Los Angeles, and Examine his most recent [unread] messages. If there is a message in this category then it will list the header information on the user console.

Similarly a command string:

```
+--------------------------+
!  #R 29, #U 18, #O <CR>  !
+--------------------------+
```

would retrieve the message 29, recover an unintentionally deleted message 18, output the message 29 text on the console and output its facsimile portion on the facsimile device.

If and when a command string does not contain a permissible combination [sequence], an error message is printed on the user console giving the details of the conflict.

3.4 The User Dialogues

3.4.1 Document Transmission:

A typical user dialogue for transmitting a one page document is shown in Fig. 3. There follows a commentary on this dialogue.

[1] Start Up:

The facsimile system software resides on a floppy-disc. On start up, typing an "L" to the Monitor loads the system into the memory, Initialises FAXSYS and prints out the title and the current version number, and asks the user his name and password to perform the Login procedure. If the login is successful it prints out the FAXSYS prompt characters "<-" and waits for a command input. For the subsequent operations, this portion of the dialogue will not be repeated.

[2] Help:

<-

If a #H <CR> is typed, a Help file is produced on the console , giving Information on the use of FAXSYS commands.

[3] Addressing:

Following the #C <CR> command the user is asked to specify the addressees, and others to whom a carbon copy of this notification and short text message should be sent. The subject field chosen should be informative as this will be the first thing the recipients will see. The information given by the user up to this point constitutes the MSG header. He is then asked the number of pages of facsimile document he wishes to send and their size.

[4] Document Feeding:

At this point the facsimile document is inserted into the machine for scanning. While the text message is being typed in, the scanned data is processed and stored on a floppy-disc.

[5] Message Composition:

The FAXSYS provides six editing characters which may be used to correct errors during this period. These are:

^A	to delete the last character
^W	to delete the current line
^N	to expunge the message
^R	to type the current line
^S	to type the entire message
^D	to end the text input.

here ^A means <CONTROL> and "A" keys pressed simultaneously etc. ^A and ^W can be used repeatedly. ^N allows the deletion of the entire text for a fresh start.

[6] Message Transmission:

Here the user types in the command string

#N, #T ISI <CR>

to indicate the transmission of his message. Since the MSG system is used directly, its facilities for immediate, delayed or queued transmission are invoked, and the whole message is sent in one burst. Simultaneously, the facsimile data is released to a message POOL on the Data-Computer and addressees' message vectors are updated to contain a new entry pointing to the message in the pool. This is discussed further in Section 4.

[7] Message Confirmation

Before the message transmission starts, the user is asked whether or not he wishes to receive a confirmation message after delivery. This is an optional facility used for mimicking "Recorded Delivery". When enabled, following the retrieval of the facsimile message the sender receives an automatically generated text mesage giving the necessary details. The use of this facility is limited to the main recipients, those specified in the "TO:" field. An example of this is given in Section 4.4.1.

[8] Message Delivery:

Quite distinct from message confirmation (7) [which implies successful message retrieval] the user is also assured that his message has been correctly dispatched. Some of these replies come directly from MSG system, and thus all of its options such as non-delivery, queued delivery, etc., may arrive. Similarly, confirmation of successful transmission of facsimile data is generated by FAXSYS when the appropriate signals are received from the Data-Computer.

DIALOGUE		COMMENTS
L <CR>		**FAXSYS** loaded
*** UCL FAX-MESSAGE SYSTEM VERSION X.04 ***		
YOUR NAME: KIRSTEIN <CR>		[FAXSYS Login
PASSWORD: <SECRET> <CR>	[1]	[Not echoed
LOGIN O.K.		
... TYPE #H FOR HELP	[2]	[User Guidance
<-#C <CR>		[Compose Message
TO: YILMAZ@BBNE <CR>	[3]	[Addressing
CC: KIRSTEIN@ISIA <CR>		[Self copy
SUBJECT: DEMONSTRATION OF UCL FAXSYS		
NUMBER OF PAGES: 1 <CR>	[4]	[Document
PAGE SIZE: A4 <CR>		[Feeding
INSERT FAX DOCUMENTS ...		
AND TYPE YOUR MESSAGE.		
TEXT MESSAGE	[5]	[this
<-#N,#T ISI <CR>	[6]	[Message
		[Transmitted
TENEX IS O.K.		[Net Status
DATA-COMPUTER WILL BE DOWN BETWEEN		
12.00 AND 14.00 ON THURSDAY 6-NOV.-1977		[From FAXSYS
MESSAGE CONFIRMATION [Y or N]: Y <CR>	[7]	[Recorded Delivery
DELIVERING MESSAGE ...	[8]	[Confirmation
YILMAZ AT BBNE ... O.K.		[of text
KIRSTEIN AT ISIA ... O.K.		

FAX IS DELIVERED		[of Fax.

Note:

Characters input by the user are underlined, and Carriage Returns are indicated by <CR>.

Fig 3. Dialogue for Message Transmission

Fig 3(a). Facsimile Message Transmitted in Section 3.4.1

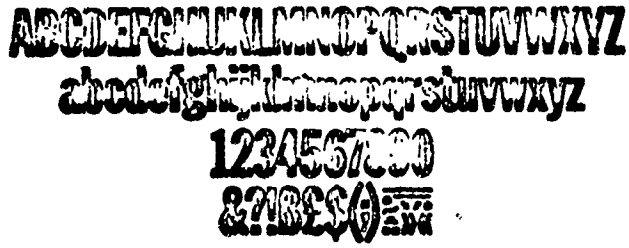


Fig 3(b). Facsimile Message Retrieved in Section 3.4.2

3.4.2 Dialogue for Facsimile Message Retrieval

Since the text portion of a facsimile message is sent via the MSG system, it can be retrieved on any alphanumeric terminal. Thus the recipient of a text message may access MSG on the Tenex which he uses for regular messages in the course of his work. The retrieval dialogue takes the form shown in Fig. 4.

After Login [1], and invoking MSG [2] he is informed about a new entry in his message file. This information which is automatically printed on the user console contains three distinct areas. Referring to the Fig. 4, these are:

[1] <Message Status><Message Number><Date><Sender>

[2] <Message Type><Message Length><Cross Reference>

[3] <Subject of the Message>

Fields [1] and [3] are generated by MSG system where [3] is an exact replica of the Subject field typed in by the sender [see Section 3.4.1. The second field in the header is generated by FAXSYS during message composition [Section 3.4.1]. This part of the header indicates to the recipient that the message is associated with a facsimile document which is 1 page long, and it is stored at CCA with the reference number 771015092551. Generation of this code is discussed in Section 3.5.

Still connected to MSG, the user may wish to see if there is any interesting text for him. He can call all the processing facilities for MSG to file, forward or delete his message as he desires. However, to retrieve the actual facsimile document, the recipient must use his facsimile terminal. An explanation of this dialogue is given below.

[1] This is the same Login to the Facsimile terminal as in Section 3.4.1

[2] The user types in the #L BBNE command to access MSG system at that site. He then receives a header listing of his unread messages.

[3] The user gives the #R 12, #O command requesting the retrieval and printing of his facsimile message. The FAXSYS then types the header of the message and asks for confirmation.

[4] When confirmed by a <CR>, the text portion of the message is retrieved and printed on his console. Simultaneously, by using the cross reference number in the text message header [CCA%771015092551] retrieval of the facsimile document is initiated.

[5] Here the #O command takes its effect, and the facsimile document is reproduced.

[6] Having retrieved his facsimile message, the user deletes the message, exits from MSG, and closes connections to the Data-Computer.

DIALOGUE	COMMENTS
L <CR>	[1]
*** UCL FAX-MESSAGE SYSTEM VERSION X.04 ***	
YOUR NAME: YILMAZ <CR> PASSWORD: <SECRET><CR> LOGIN O.K.	
... TYPE #H FOR HELP	
<-#L BBNE <CR>	[2] Connect to MSG
MSG --- Version of 1 April 1977	
-+ 12 15 Oct Kirstein at ISIA <<FACSIMILE, PGS=1, ID= CCA%771015092551>> DEMONSTRATION OF UCL FAXSYS	
Last Read: 14 Oct 1977 12 msgs, 7 disc pages	
<-#R 12, #O <CR>	[3] Retrieval
<<ID=CCA%771015092551>> [Confirm] <CR>	
Mail from ISIA rcvd at 15 Oct 1977 0957 Date: 15-Oct-1977 From: Kirstein at ISIA To: Yilmaz at BBNE cc: Kirstein at ISIA Subject: <<FACSIMILE, PGS=1, ID= CCA%771015092551>> DEMONSTRATION OF UCL FAXSYS	[4]

```
----
[TEXT MESSAGE]

----

Retrieval of CCA%771015092551 is in progress ...  O.K.

-----                                                    [5]
FAX Message Printed !

<-#D 12, #B <CR>                                         [6]

DELETING CCA%771015092551  [Confirm] <CR>
```

Fig. 4 Dialogue for retrieving facsimile Messages

3.5 Generation of Message Code Numbers

Since the textual and facsimile information are stored on different Host computers, it is necessary to devise a cross-referencing mechanism, so that messages can be manipulated uniquely. This is achieved in the following way:

During the Login procedure to a Tenex, FAXSYS obtains the time and date of login, which has the format :

DAY - MONTH - YEAR, HOURS - MINUTES - SECONDS

[e.g. 15 - Oct - 1977, 09 - 25 - 51]

This information is rearranged to form a twelve digit number which is unique to the message being sent. For the example given above the code number becomes : 771015092551

In fact, this code serves two purposes. Firstly as a link between the two portions of a facsimile message, and secondly as an identifier to store the facsimile information on the Data-Computer. However, since the Data-Computer does not accept a number as the first character in file names, the above code is padded with "CCA%", thus producing : CCA%771015092551. This identifier is generated and added into the text message headers before transmission. Its use during the storage and retrieval of facsimile data is discussed in the next section.

4 Storage and Retrieval at CCA

4.1 Introduction

The Data-Computer [Ref. 3] is a large-scale data storage utility offering data storage and management services to other computers on ARPANET. The system is intended to be used as a centralised facility for archiving data, for sharing data among the various network hosts, and providing inexpensive on-line storage. The Data-Computer is implemented on dedicated hardware, and comprises a separate computing system specialised for data management.

Logically the system can be viewed as a "closed box" shared by multiple external processes, and accessed in a standard notation called "Data-Language" [Ref 3]. The system is provided with an Ampex Terabit Store of 10**12 bits. While this is mainly used for seismic data, it does contain software and hardware ideal for archival storage. Thus it is an excellent vehicle for our experiments.

In a paper of this nature it is not possible to enumerate every aspect of the Data-Computer, and our use of its facilities. Therefore, the following sections are limited to a discussion of the general principles closely related to our application, and the details are kept to a minimum.

In Section 4.2 we discuss the directory structures set up in the Data-Computer. Section 4.3 presents the introductory concepts of file security and password organisation. Section 4.4 describes the data structures used for storing facsimile information. And finally in Section 4.5 we discuss the mechanism for accessing the Data-Computer, and translation of user requests into Data-Language.

4.2 Directory Structure

All data, whether being transmitted to or from the Data-Computer, or stored within it, must be formally described to the Data-Computer. These data descriptions are kept in the Data-Computer directory which has a tree structure as illustrated in Fig. 5. The entries in this directory are called "nodes", and only the bottom-most nodes of any branch can contain data descriptions. There are three kinds of directory nodes. These are:

FILE: containing a description of the format in which the data is stored within the Data-Computer.

PORT: Containing a description of the format in which the data is to be transmitted to and from the Data-Computer.

NODE: A node which does not contain data description, and may have zero or more subordinate NODEs.

For example in Fig. 5, FACSIMILE, ISI, UCL and BBN are the NODEs defining the storage hierarchy of the facsimile files. FAX, USER1, USER2, etc. are the FILEs containing the data, and FAXP is a PORT which defines the transmission format of the facsimile files to and from the FILE FAX.

Each element in the directory has a unique name which enables references to any node within the directory structure.

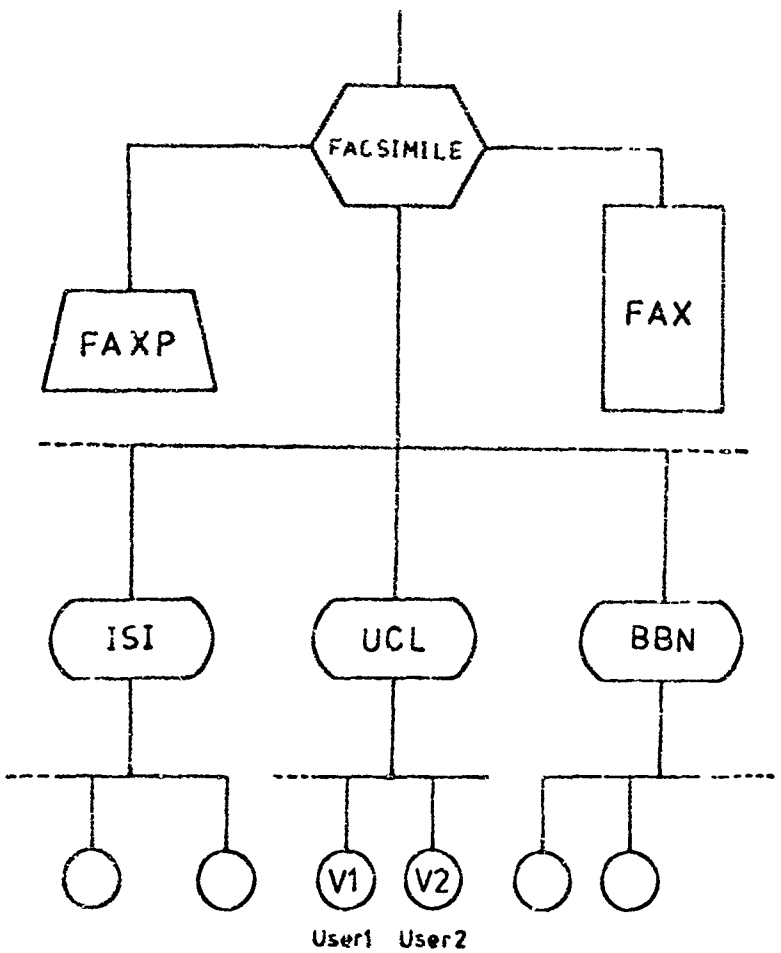


Fig 5 Facsimile Directory Structure at CCA

4.3 File Security and Passwords

The Data-Computer provides restricted access to FILEs and PORTs by means of privilege blocks which define a class of users and set of privileges to be granted to such users. A privilege block is fully specified during the creation of FILEs and PORTs, and can contain the following:

User Name
Host Number
Socket Number [See Section 4.5]
User Password
Control: Read, Login, Write, Append Privileges

Whenever a user attempts to access a node, FILE or a PORT, the Data-Computer scans the privilege blocks of all the nodes along the specified path name to determine what privilege should be allowed.

Referring to Fig. 5, if certain sites desire to be particularly secure, they may set a site password at that point in the path. In our implementation password protection is given to the users at the user nodes only. Thus to access a file in USER1, a user whose mail box is at ISI, and whose password is SECRET would require the Data-Language command :

```
LOGIN FACSIMILE ('PASSWORD');
LOGIN %LOGIN.UCL.USER1('SECRET');
```

If extra security is desired at the site level, say at UCL with the password LONDON, then the second line in the above example becomes:

```
LOGIN %LOGIN.UCL('LONDON').USER1('SECRET');
```

Access restriction by means of passwords is only a part of the picture. A user who has been granted the LOGIN privilege to the UCL node may well be denied say, READing, WRITEing, or APPENDing to the files contained in USER1 . Through this selective access control mechanism a user can allow others to READ his messages but not WRITE into them, if he so desires. Further details of this topic may be found in Ref. 3.

4.4 Data Structures

There are two main file spaces involved in storing facsimile information on the Data-Computer. Of these, FAX

in Fig. 5, is the message POOL which contains a single copy of all the messages sent to the Data-Computer. V1, V2 etc. in the same figure are the user message Vectors which contain a list of pointers to the user messages stored in the pool, FAX. The file spaces FAX, V1, V2 etc. consist of hierarchial structures of containers modelling the data as it is stored on the Data-Computer. These containers hold either a piece of information or more containers as illustrated in Fig. 6. Each container in a file space is associated with a unique name, data type, size and format information, which are specified during file creation [Ref. 3].

FAX: Fig. 6(a) is a model of the storage area FAX where all the facsimile messages are kept. This file contains a list of message items called MSG1, MSG2 etc. which are the formatted versions of the facsimile messages sent to the Data-Computer. Each message space is divided into two areas, of these, USERLIST contains a list of Vector names representing the users to whom the message was sent [e.g. <UCL.PTK>, <BBN.SY>, ...], and the second, FAXDATA, contains the actual facsimile message. The FAXDATA space is further divided into two areas. The HEADER contains the reproduction parameters [Section 6], and BINDATA contains the binary data associated with the message. It should be noted that although the message containers are identical in format, the amount of data stored in the corresponding sub-containers can be different, and the information in these containers can be accessed and manipulated separately.

Message Vectors: Every user of the Facsimile System is assigned a private file space called the "Message Vector". As illustrated in Fig. 6[b], a Message Vector consists of a list of cascaded containers which accommodate the following parameters :

Message Identifier
Message Status
Message Originator Name [i.e. the Sender]

The Message Identifier is a fixed length pointer which contains the name of the message stored in FAX. The message Status flags are used to indicate

whether or not a given message is Active/Read/Deleted/ etc. The container which holds the sender's name is an optional parameter, and it is used for message confirmation as detailed in the next section.

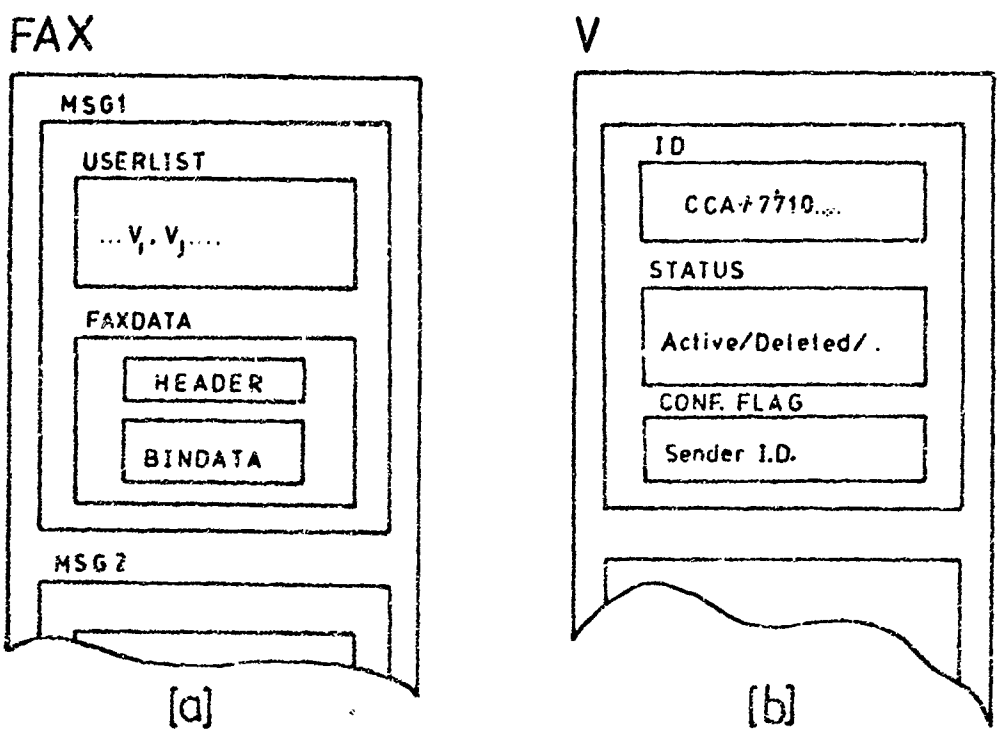


Fig. 6 Structures of FAX and V Files

4.4.1 Operations on FAX and V files

There follow a few examples of the way in which the user commands discussed in Sections 3.4.1 and 3.4.2 operate on the data structures at CCA.

#T [Transmit Message]

The "File Descriptor" and binary data associated with the message are stored in a new message container in FAX, and a USERLIST is composed specifying the names of the recipients. At the same time a new list member is added to each recipients' Message Vector which contains the message name, status [unread initially], and its originator. The latter will be set only on a confirmation request from the sender, otherwise it contains a null string.

#R [Retrieve Message]

First the user is connected to his message Vector using his specified password. With the reference number obtained from the text message, the corresponding facsimile file is located and sent to his terminal. At the end of the retrieval process the Originator field is inspected for a possible confirmation request. If this field contains anything other than a null string then, using MSG an automatically generated text message is sent to the Originator, thus confirming the successful delivery of the message. For the example given in Section 3.4.1 the confirmation message takes the form shown below.

To:Kirstein at ISIA
cc:
Subject: Delivery confirmation of CCA%771015092551

Facsimile message CCA%771015092551
sent to SY at BBNE
on 15-Oct-1977
successfuly retrieved on 15-Oct-1977
at 16:20 GMT

#F [Forward Message]

Forwarding a message to other users follows very much the same lines as for #T, except that with #F

command the USERLIST of an existing file is updated to contain the additional recipients' names.

#D [Delete Message]

The user pointers in his message Vector and USERLIST in FAX are removed. Note that a user cannot destroy a message which has multiple recipients. Permanent removal of a message occurs only when the USERLIST of a given message becomes a null string [i.e. when the message is marked for deletion by all the recipients].

4.5 Data-Computer Access

The Data-Computer is designed so that the external computers may transfer data to and from it by a two-stage process. First they connect to one stream in the system by the standard Initial Connection Protocol of ARPANET followed by a Login. This connection is kept open throughout the data transmission operations. It is used to control the data transmission and set up appropriate transmission parameters: we call this the Control Stream [CS]. When the control stream has been set up, and the appropriate path to the user node established, a second Data Stream [DS] is opened by CS. All data transfers take place through this secondary connection. Data transmission over CS is also possible, but this option is limited to alphanumeric information and it is far less efficient.

Figure 7 shows an example of the commands required to Log in and retrieve the file CCA%771015092551 sent to Kirstein at ISIA. It is assumed that the retriever is using a TIP and wishes the data to come onto PORT 262146 of the TIP. Also to simplify the dialogue it is assumed that the FAX file contains only one message file and the Message Vector is not used. In this example PASSWORD and SECRET are needed to access FACSIMILE and KIRSTEIN nodes, and it is assumed that the ISIA node does not require a password.

The Data-Language [DL] is a clearly defined dialogue, with numeric portions facilitating computer rather than human processing. Fig. 7 shows that it is possible to drive the Data-Computer from a keyboard terminal. If the character ports on the TIP are used, two terminals will be required; one for Data-Language [CS], and the other for data [DS]. Although the Data-Language appears to be very awkward and verbose for a human user, its responses are

ideally suited to synchronising activities with other computers or intelligent terminals such as ours.

```
 H 31<CR>                                          ] Connect to D-C
 R F S 203 <CR>                                    ] Socket 203
 I C P <CR>                                        ] ICP
Trying ...

Open

;0031    771015092551    IONETI:    CONNECTED TO LONDON TIP 1200000
;J150    771015092559    FCRUN:     V='OC-1/01.13' J=1 DT = 'MONDAY',
                                    OCTOBER 15, 1977 08:59:26-est
;0041    771015092933    ONCINX:    DATACOMPUTER GOING DOWN IN 60 MIN ..
;J200    771015093021    RHRUN:     READY FOR REQUESTS
.I210    771015093022    LAGC:      READING NEW DL BUFFER

LOGIN FACSIMILE ('PASSWORD'); <CR>                 ] Connect
                                                   ] to FACSIMILE NODE
.I210   ...                                        ] Message
OPEN FAXP; CONNECT FAXP 262146; <CR>               ] PORT FOR
                                                   ] DATA TRANSFER
.I210                                              ] Ready.  SYNC
OPEN %LOGIN.ISI.KIRSTEIN('SECRET').CCA%771015092551; <CR>
                                                   ] Opens file

.I210                                              ] SYNC

FAXP = CCA%771015092551; <CR>                      ] Send Data

CLOSE CCA%771015092551; <CR>                       ] User tidies

DISCONNECT FAXP; CLOSE FAXP; <CR>                  ] File and Port
```

Fig. 7 Data Language Commands for Retrieval

5 Experiments with XGP

5.1 The XGP System

The Xerox Graphics Printer [XGP] at ISI is the complementary part, FAX2, of the facsimile system depicted in Fig. 1. This device operates in a "raster-scan" fashion, and it is driven by a PDP-11-TENEX combination. Several sites on ARPANET use the XGP, with slightly different configurations, to produce formatted, high quality documents. For this reason, a considerable body of software has been developed [Ref. 6] to drive the XGP for text oriented applications. Since the XGP hardware operates on "dot-matrix" representations of arbitrary character sets, we have developed an interface to the existing XGP software at ISI, and devised a suitable character set to print facsimile images on the XGP. There also exists a facility [Ref. 7], provided by CCA, for TENEX-Data Computer communication. The combination of hardware and software used for our application is mainly derived from those which already existed; therefore the details of this system [Ref. 5] will not be given here. In the following sections we discuss the techniques which we have used for KD 111-XGP communication, with the aim of achieving device independent facsimile transmission. Section 5.2 emphasizes the need for machine independence, and in Section 5.3 we show how this was accomplished in the case of XGP- KD 111 communication.

5.2 A Network Virtual Facsimile Terminal

In the system of Fig. 1, it is important to stress that we wish to transfer an image from one machine [the KD 111] to another [the XGP] with different characteristics. The principal differences are: the form of data [analogue versus digital], speed, synchronisation, and form of printing. These differences are by no means peculiar to these two devices. Almost invariably, all present day facsimile equipment suffers from this kind of incompatibility to a greater or lesser degree. Although there is a considerable urge to standardise facsimile equipment [Ref. 8], it is most likely that the products of different manifacturers will always have their own peculiarities. For the kind of applications envisaged here, we believe that, within limits, one way of overcoming these incompatibilities is to define a "Network Virtual Facsimile Terminal" which allows local mapping of facsimile information into real terminals. In this way, data is transmitted and stored in a canonical form, then retrieved and mapped into the local terminals' requirements. With

this approach it is then possible to accommodate a wide range of equipment in a given environment, such as ARPANET. The fundamental problem in mapping data from one device to another is the possible resolution differences that may exist between them. Unless these differences are within a tolerable range, it is not possible to preserve image integrity. For most applications, however, a proportional reduction (or expansion) of the original image may not be so severe as to produce unacceptable results. If this transformation can be lived with, then the problem of defining an NVFT reduces to that of self-descriptive representation of facsimile information. In this context, we have taken the view that a "scan-line" is a more meaningful entity than the constituent elements (pels) of the image, and have adopted data structures based on "scan-lines" as the basic units of facsimile information. Fig. 8 shows a list of the parameters contained in the HEADER of facsimile files stored on the Data-Computer.

B :	1	Byte	Transmission Byte Size
L :	3	"	Number of Bytes per Scan-Line
S :	5	"	Number of Scan-Lines in the File
G :	1	"	Grey Levels Used [e.g. 2 for Black and White]
H :	2	"	Horizontal Resolution of the Originating Device
V :	2	"	Vertical Resolution of the Originating Device
I :	1	"	Compression Process Identifier

Fig. 8 HEADER Parameters for Facsimile Files

5.3 KD 111 to XGP Mapping

Scan-Lines:

With an analogue device, such as the KD 111, determination of the individual scan-line boundaries is a necessary operation for achieving device independence. At a sampling frequency of 2.4 KHz., our usual clock rate, a scan-line does

not contain an integral number of bits. However, from the scanning time in bits, we have derived the exact number of bits for each scan line, and rounded these off to a fixed length (105-8 Bit bytes or 840 bits). With this truncation process timing can be in error at most by 1 bit width as illustrated in Fig. 9(b). This was obtained by simulating the output on a character terminal using one character per bit. It is interesting to note that if the fractional bits were allowed to accumulate, then an inaccuracy of 0.1 bits/scan-line would result in a sheer of 12 degrees by the end of an A4 size page. The S,L and B parameters given in the HEADER (see Fig. 8) are the result of the above transformation process.

Resolution:

The resolution of XGP is twice that of the UCL KD-111 [96 pels/inch] at the usual sampling rate of 2.4 Kbps. This difference was accounted for by the facsimile character used. The XGP system is designed to operate on 7-bit character sets, but many of the first 63 are used for control purposes. Limited to the use of 6-bits, the facsimile post-processor at ISI maps each 6-bits of the KD-111 data into 24-bit representation of the facsimile character set. This is illustrated below:

Bit pattern generated by the KD-111	101101
Representation in the character set	1001101
Bit pattern sent to the XGP	110011110011
	110011110011

Although this transformation overcomes the resolution difference, at the time of these experiments [Ref. 5], the PDP-11 software was not quite able to handle dense facsimile characters. Fig. 9 (a) shows the output of a test pattern on the XGP that was originated from the UCL KD 111.

Fig. 9(a) Sample XGP Output

Fig. 9(b) Sample Output Simulation

6. CONCLUSIONS

The system described in this paper is operational, and provides a valuable tool for integrating facsimile transmission with message and document services. While the system is reasonably easy to use, it has highlighted several areas where further work is required.

1] Clearly a digital facsimile device with a controllable digital interface, such as DACOM 450, would be more appropriate.

2] The role of a backing store with such a digital device would require re-examination.

3] The operational characteristics would be simpler if the message facilities were integrated with the storage system. However this integration is not necessary.

4] The inclusion of encryption would be fairly straightforward, following the same technology as for any other digital data. Because of the way notification is done, the notification messages could contain key information about the encription of the facsimile files. This aspect has not been studied in depth by us.

5] There is no inherent reason why this system cannot work with a wide range of facsimile terminals, and data representations. For more complex transformation, dedicated uP should be used for this function.

6] The present experiments have been with a facsimile device attached via an EIA 232 interface and asynchronous transmission format (with synchronous clocking). Future devices would be better served with an appropriate synchronous communication discipline, such as X25. There is nothing in the technology which requires packet switched rather than circuit switched communication systems.

7] The economics of the system must be studied much more carefully. A key aspect is the method and tariff structures for providing the storage and message processing functions.

8] The present experiment is the first we know which has used Terabit stores as ternary storage media. The adequacy and economics of this form of storage must be studied further.

9] The system concept is of recipient oriented interrogation of this message system. In view of the small size of the message involved , we must investigate further the advisability and economic feasibility of unsophisticated notification of facsimile file availability.

We believe that this type of development is a foretaste of the type of integrated message, word-processing and facsimile systems that will shortly become available.

#A	Attach to MSG System	None
#B	Break Network Connections	None
#C	Compose a Facsimile Message	None
#D	Delete Message	Msg. No.
#E	Examine Recent Messages	Host I.D.
#F	Forward Message	User List
#H	Help, User Guidance	None
#K	Kill Deleted Messages	None
#L	Login to MSG System	Host No.
#N	Network Status	Host I.D.
#O	Output Facsimile Message	None
#P	Print; to examine	None
#Q	Quit from MSG System	None
#R	Retrieve Message	Msg. No.
#T	Transmit Message	None
#U	Undelete Message	Msg. No.

Table 1 FAXSYS User Command Summary

ACKNOWLEDGEMENT

We acknowledge the help of the British Library Research and Development Department, under Grant SI-G-172, who supported this work. It would not have been possible without the support of the Ministry of Defence [Contract AT/2047/064], the Science Research Council [Grant B/RG/67022], and the US ARPA, under the Office of Naval Research [N00014-77-G-0005], who made the use of ARPA Network facilities possible. Finally, many colleagues at the Computer Corporation of America, the Information Sciences Institute, and the UCL INDRA group helped in the work.

REFERENCES :

1. Roberts L.G. et al, "The ARPA Computer Network", Comp. Comm. Netw., (Prentice Hall, Englewood Cliff, N.J. 1973) pp 485-500

2. Holg C, "The ARPANET/TENEX Primer and the MSG Message Handling Program", University of Southern California, May 1977

3. --- , "Datacomputer Version 1, User Manual", CCA Working Paper No. 11, Computer Corporation of America, August 1, 1975

4. Currier R, et al, "ISI XGP System", University of Southern California, Information Sciences Institute, July 1975

5. Kirstein P.T., "University College London ARPANET Project Annual Report", INDRA TR-27, University College London, February 1976

6. Newcomer J., et al, "An Introduction to XOFF" Carnegie Mellon University, 1973

7. Farrell J., "A Program to run the Datacomputer" CCA User Information, Computer Corporation of America, April 1975

8. CCITT S and F Working Group, "Report of the 'ad hoc' group on aspects of Question SF 14-Facsimile Oslo, October 1975

PAPER 3

FACSIMILE TRANSMISSION IN MESSAGE PROCESSING SYSTEMS WITH DATA MANAGEMENT

BY

Peter T. Kirstein and Sinan Yilmaz
Department of Statistics and Computer Science
UNIVERSITY COLLEGE LONDON

ABSTRACT: The methods of message processing, which have been developed recently for textual messages, are extended to handle facsimile information. The paper discusses briefly a UCL experimental system using ARPANET, a Message Processor, an Archival Data Management System and intelligent digitised facsimile terminals. The components in the costs are analysed via a case example and proposals made to achieve an economically viable system.

1. INTRODUCTION

Experience with ARPANET (Reference 1) has shown that an extremely important proportion of its use is for Message Processing. The message processing facilities being developed are very much more sophisticated than Telex. They include facilities for transmission to multiple addressees, archiving of messages and complex filing facilities by the recipient. The facilities found useful have been discussed elsewhere (Reference 2). The ARPANET experience has been validated by the uses to which most commercial Time Sharing Systems have been put. In most cases the uses have developed in an ad hoc and undercover way because regulatory consideration precluded a service offering. Recently at least one U.S. regulated Value Added Carrier has announced an official general public system (Reference 3) and others have sold sytems for private use inside corporations (Reference 4).

There is a strong trend towards capturing data at source into coded machine-readable form. In this form a typical A4 page would contain about 2.5K characters or 20K bits. However, there is a class of documents which are not produced in-house, or which may contain information not readily representable in ASCII characters. This class of information can be transmitted only by some facsimile transmission method. In facsimile form the information is much more diffuse; an A4 page at normal scan rates (only 2 level) would require up to about 2M bits. However, appropriate forms of coding would allow such data to be compressed to 200K bits. Although the information is an order of magnitude less dense than coded text transmission, there are many cases where this form of data storage and transmission is the only feasible one.

The purpose of a recent UCL research project has been to explore how facsimile techniques can be integrated into sophisticated message processing systems. Our approach has been to note that a message has two portions: its control description and its data. The control description includes its subject matter, identification name, length, date created, desired addressees, originator, length etc. The data of a message is the actual textual information. The control description or header portion is normally short and

highly standardised in form. In computer based message systems it is designed to be easily comprehended and automatically processed by computers. The data in messages is usually an arbitrary text string, though it may contain control characters necessary for outputs on special devices (e.g photocomposition devices, Reference 5) or for special formatting (e.g. in text processors, Reference 6). Facsimile data can be manipulated in the same way as textual data. The header portion is in identical form; the data portion is now coded scan-lines rather than coded alphanumeric characters.

With text messages, there has been a controversy as to whether the text data should be sent to each recipient, or only the headers and pointers to the data should be sent and just one copy of the data be stored. Many message systems on ARPANET use the former (e.g. Reference 7); some use the latter (e.g. Reference 8), particularly when they are designed for long documents. Similarly many commercial systems use the former (e.g. Reference 3), others the latter. Distributed systems like that of Reference 4 use a combination of the two. The data is transmitted only once to each destination system, but is stored there in one copy for each addressee. With facsimile data, which is much denser than coded alphanumeric, it becomes particularly important to avoid multiple transmission and storage.

An early problem encountered even with alphanumeric terminals is the incompatibilities between terminal types; these include character code (e.g. EBCDIC and ASCII), line length, use of control characters etc. One way of overcoming these incompatibilities is to define real standard terminals: the International Alphabet Number 5 and ASCII are the result of this process. A second technique is to define a Network Virtual Terminal (NVT, References 9 and 10) which allow local mapping to a real terminal. Many commercial data networks go some way to adopting an NVT approach. In much the same way it is possible to define an idealized Network Facsimile Terminal (NFT) whereby data is transmitted and stored in an appropriate form, to be retrieved and mapped into local terminals. Through this process it is then possible to support different facsimile terminals within a given system.

In general it may be necessary to update periodically the control procedures for facsimile processing. These procedures can either be updated locally in the facsimile terminals, or centrally in a Network Access Machine. One of the UCL activities has been the investigation of the Network Access Machine in this role (Reference 11).

In Section 2 we will give a brief overview of the principles of the UCL Facsimile Service System. In Section 3 the UCL Facsimile terminal and system will be described in greater detail.

The UCL experiment was performed in a specific environment with specific service facilities. An economic service would require the existence of standard terminals and services of an appropriate kind. In Section 4 we discuss what these services should be, and then consider the characteristics required in the terminal.

The UCL approach is potentially applicable in any data network environment. Tariff structures have now been announced for many of the present and planned data networks. The potential economics of the services envisaged will be discussed in Section 5. Finally, in Section 6, some conclusions are drawn about the potential application of the ideas presented here.

2. The System Overview

In principle the system we have developed is shown in Figure 1. An intelligent facsimile terminal FAX1 is connected in to a Data Network DN via a suitable serial interface. Also attached are: an Archival Data Base Management system, ADMS. a message processing system MP and another intelligent facsimile terminal FAX2.

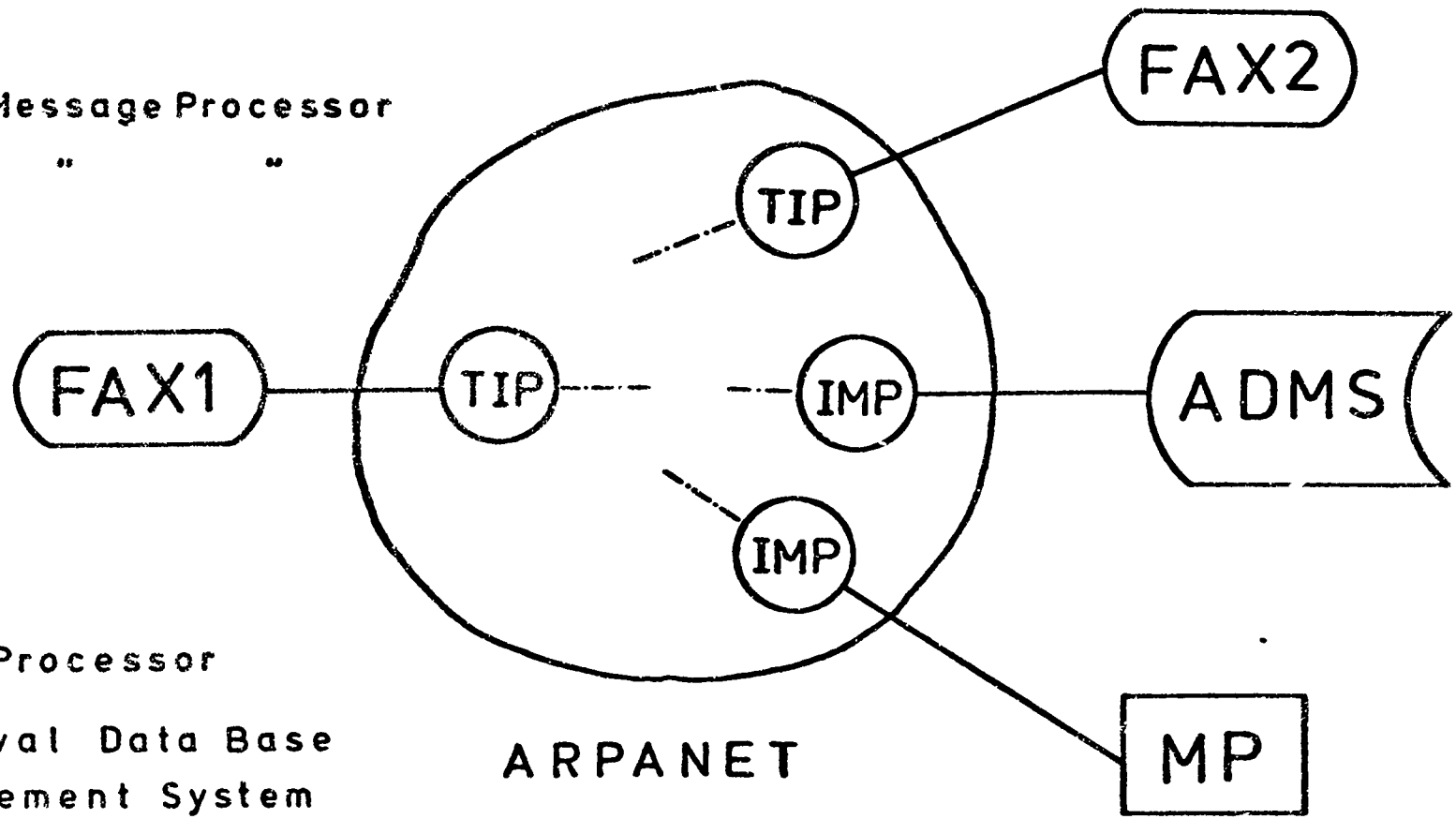
MP and ADMS in Figure 1 are shown in different places, but there is no reason why they could not be accommodated within the same host. Our aim is to develop a system where the facsimile terminal is sufficiently intelligent that it can read documents with a text header and address list, to transmit and store them in ADMS while notifying the addressees through MP that the document is stored in ADMS.

Later the addressees may access ADMS and retrieve the file on their own facsimile device FAX2. At the recipients' option (or possibly the sender's, to mimic 'recorded deliver'), each addressee may automatically inform the originator that the document was received. In addition he may wish to store the facsimile data in a canonical form, so that it may be retrieved from different facsimile devices.

There is an important difference between the storage of facsimile and short text messages. Such textual information is usually small (less than 500 bytes), whereas the facsimile data is much larger, typically 25K bytes for an A4 size page with an optimum data compression. Hence the textual data can be stored in multiple copies, one for each addressee in his "mailbox". The facsimile information is stored only once in ADMS, and the path name to retrieve it is known by the addressees from their text messages.

Some document processing systems use similar techniques; the determination of the filing, storing and retrieval facilities is considered to be beyond the scope of this paper.

In our attempt to implement this system we have used ARPANET as the data network. For FAX1 we attached a microprocessor to an analog 4-6 minute facsimile device (Section 3). This was then connected



IMP : Interface Message Processor

TIP : Terminal " "

MP : Message Processor

ADMS : Archival Data Base Management System

FAX1, FAX2 : Intelligent Facsimile Terminals

Fig.1 Overview of an Idealised Facsimile System

through an asynchronous link to ARPANET. For the MP of Figure 1, we use DEC/TENEX, usually at the University of Southern California.(ISI), supporting the message system MSG (Reference 7). The TENEX is a development of the PDP 10 with special hardware paging and an appropriate operating system, which is widely available on ARPANET. For ADMS, the Data Computer at the Computer Corporation of America (CCA), with an on-line storage capacity of 10^{12} bits, was the natural candidate and has been used exclusively. As for FAX2 we are using the XGP printer at the Information Sciences Institute at the University of Southern California (ISI), which is controlled by a DEC PDP-11/ TENEX configuration.

In this implementation the control of data flow, (including the set up, supervision, and close down of virtual calls) was performed by the microprocessor in FAX1. This procedure has the disadvantage that any changes in the operation of MP or ADMS must be reflected into the driving software for all the facsimile devices. An alternative approach is to have this control provided as a network service.

An independent research project has established a Network Access Machine (NAM) (Reference 11). This is an excellent application of NAM, namely to use it to automate the functions of controlling the connections to the Data Computer and the Message Processor. For the purpose of demonstrating one specific facsimile system implementation, the NAM does not provide any functionally superior facilities. However, system maintenance, particularly if there are many facsimile devices deployed or different MPs used, would be far easier by NAM techniques. The rest of this paper will ignore details of the NAM techniques.

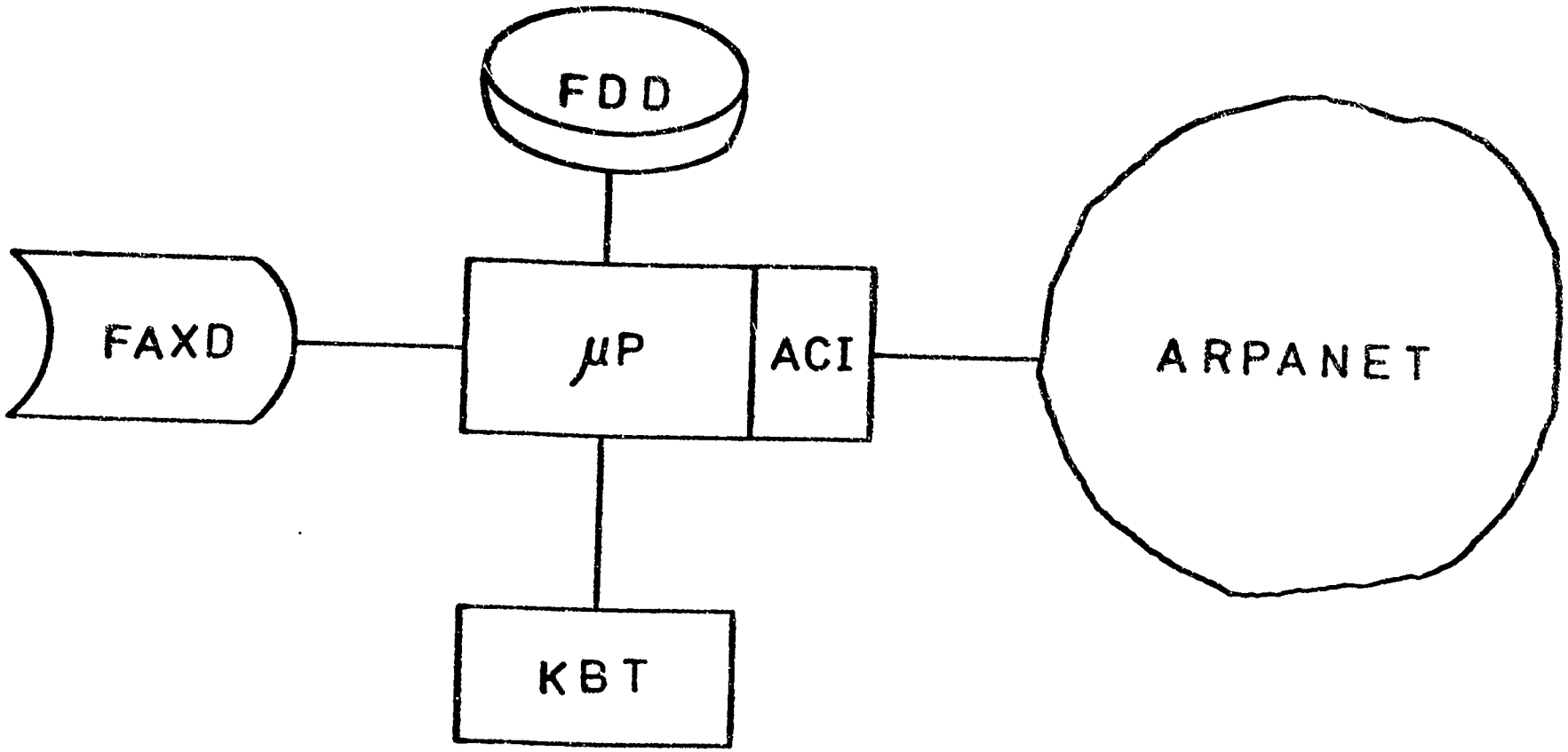


Fig. 2 UCL Facsimile Terminal

3. UCL FACSIMILE TERMINAL AND ITS USES

3.1 Introduction

The system outlined in Section 2 can be described at many different levels. In this section we will describe the UCL facsimile terminal itself (Section 3.2) and give examples of its use in message transmission. The examples given contain two user dialogues. One for message transmission (Section 3.3.1) and the other for message retrieval (Section 3.3.2).

The message transmission dialogue is mainly concerned with the composition of a message file (containing textual and facsimile information), its local verification, its submission to the MSG system and the Data Computer and confirmation of delivery. This dialogue also illustrates the binding of linkages between the two portions of the message and the supervision of message transmission which are almost transparent to the user.

The retrieval of a message can have either one or two stages. Because the notification comes via the text message system MSG, the user may receive notification of facsimile messages during straightforward text retrieval from a conventional terminal. In this case the retrieval of the facsimile portion must be a separate exercise. Alternatively, as illustrated in the retrieval dialogue, the user may access the MSG system via the facsimile terminal. In this, somewhat simpler, case the notification and retrieval become an integrated operation.

3.2 The UCL Facsimile Terminal

Figure 2 shows the basic components of the facsimile terminal we have developed. Physically it consists of Plessey KD-111 4-6 minute analog transceiver, a 24K byte INTEL 8080 microprocessor (uP), a floppy disk (FD), a keyboard terminal (KBT) and a communication interface (ACI).

The KD-111 analogue output was digitised with two level threshhold logic, and interfaced by a standard V24 interface to the uP; the output was sampled at 2.4Kbps, giving about the resolution of the fundamental device. Conventional analogue devices have an end-to-end synchronisation phase, and must then pass data synchronously. Most packet data networks (and ARPANET is no exception) cannot guarantee to deliver data at any particular constant rate. For this reason data must be staged on to a floppy disk ; the disk can hold 4-5 pages of facsimile data. The keyboard is needed to control the system. It provides a medium for the user to compose his text message, specifying addressing information, and to initiate transmission and retrieval of messages. In our system a standard teleprinter is used, but a simple keyboard would be adequate. The teleprinter is useful for noting the fate of facsimile transmission; if hard copy is not required, a CRT is probably better. The facsimile output device, or a very cheap printer, can be used also to meet the record-keeping requirement.

Because we wished to access the network via a standard terminal interface, we used a V24 modem interface, with asynchronous data format, for its communication interface. The flow control, which was still required over the ACI, was best done in some block mode; a future version of this terminal will use the X25 communication interface.

The microprocessor in Figure 2 provides the intelligence of the facsimile terminal. Intelligence is required to free the user from lengthy routine operations that are inevitable when such devices are used in the computer networks. It is required also to control the staging of data via the disk, to control the communication port, to sequence all operations correctly, to interpret user commands and to inform the user of the progress of his commands. At present the uP also provides data compression and decompression. In a future system, some of these functions would be accomplished on dedicated uPs.

3.3 The User's View of the System

The user FAXSYS dialogue takes place via the system console. A set of commands permits the user to interact with the user interface process. Input of a command causes this process to activate the associated segments to perform the required operation. Although the user is informed about the progress of his request, he is in no way made conscious of the complex operations behind the scenes. In fact, our earlier work on a similar implementation (Reference 12) had shown the importance of providing the user with an understandable and unified view of the system. A typical dialogue would Compose, Review and Transmit a transaction. In the composition phase, an address, a subject matter and a text message are input; simple editing is permitted during composition. The facsimile document is also read in. In the review phase (if desired), the message and document are output locally, (if desired). In the transmit phase, the facsimile portion is sent to the Data Computer; also a message, including both the text message and a pointer to the location in the Data Computer, is sent to the addressees via the message system. A typical dialogue is shown in Figure 3.

The text message can be retrieved in the standard way from the message system with an alphanumeric keyboard. Additionally (or alternatively) both the text and the facsimile portion can be retrieved by the UCL Facsimile Terminal. Here a typical dialogue is shown in Figure 4. A fuller description of the operation of the system and all the facilities is given elsewhere (Reference 13). A list of FAXSYS commands is given in the Appendix.

In Figure 4, the facsimile data was transmitted to the Data Computer (DC). During the resulting dialogue (invisible to the user) a pointer to the facsimile file (CCA%771015092551) was generated by the uP, and added to the text message. This pointer was used autonomously in the retrieval of Figure 5 to locate the facsimile file. The ID number is generated from the time-stamped synchronisation messages sent by the Data Computer (year, month, day, hour, minute, second). Hence it is unique.

DIALOGUE	COMMENTS
<CR>	(FAXSYS Loaded
*** UCL FAX-MESSAGE SYSTEM VERSION X.04 ***	
YOUR NAME: KIRSTEIN <CR>	(FAXSYS Login
PASSWORD: <SECRET><CR>	(Not Echoed
LOGIN O.K.	
... TYPE #H FOR HELP	(User Guidance
←#C <CR>	(Compose Message
TO: YILMAZ @ BBNE CR	(Addressing
CC: KIRSTEIN@ISIA <CR>	(Self Copy
SUBJECT: DEMONSTRATION OF UCL FAXSYS	
NUMBER OF PAGES: 1 <CR>	(Document
PAGE SIZE: A4 <CR>	(Feeding
INSERT FAX DOCUMENTS ...	
AND TYPE YOUR MESSAGE.	
(Message)	(Text Message
	(Terminator
←N.#T#ISI <CR>	(Message
MESSAGE CONFIRMATION (Y or N): Y <CR>	(Recorded Delivery
DELIVERING MESSAGE ...	(Confirmation
YILMAZ AT BBNE ... O.K.	(of text
KIRSTEIN AT ISIA ... O.K.	

FAX IS DELIVERED	(of Fax.
←	(Next Command

Note:

Characters input by the user are underlined, and Carriage Returns are indicated by <CR>.

Figure 3 Dialogue for Message Transmission

DIALOGUE	COMMENTS
CR	(Login
*** UCL FAX-MESSAGE SYSTEM VERSION X.04 ***	
YOUR NAME: YILMAZ < CR >	
PASSWORD: < SECRET ><CR >	
LOGIN O.K.	
...TYPE H FOR HELP	
← #L BBNE <CR>	(Connect to MSG
MSG --- Version of 1 April 1977	
-+ 12 15 Oct Kirstein at ISIA	
<<FACSIMILE, PGS=1, ID=CCA%771015092551>>	
DEMONSTRATION OF UCL FAXSYS	
Last Read: 14 Oct 1977 12 msgs, 7 disc pages	
← #R 12,	(Retrieval
<<ID=CCA%771015092551>>(Confirm) <CR>	
Mail from ISIA rcvd at 15 Oct 1977 0957	
Date: 15-Oct-1977	
From: Kirstein at ISIA	
To: Yilmaz at BBNE	
cc: Kirstein at ISIA	
Subject: <<FACSIMILE, PGS=1, ID=CCA%771015092551>> DEMONSTRATION OF UCL FAXSYS	
(Message)	
Retrieval of CCA%771015092551 is in progress ...O.K.	
-----	(Output

FAX Message Printed!	

Figure 4 Dialogue for Retrieving Facsimile Messages

3.4 Storage and Retrieval at CCA

The Data Computer is a large-scale data storage utility offering data storage and management services to other computers on ARPANET. The system is intended to be used as a centralised facility for archiving data, for sharing data among the various network hosts, and providing inexpensive on-line storage. The Data Computer is implemented on dedicated hardware, and comprises of a separate computing system specialised for data management.

Logically the system can be viewed as a "closed box" shared by multiple external processes, and accessed in a standard notation called Data Language. The system is provided with an Ampex Terabit Store of 10^{12} bits. While this is mainly used for seismic data, it does contain software and hardware ideal for archival storage. Thus it is an excellent vehicle for our experiments. In a paper of this nature it is not possible to enumerate every aspect of the Data Computer, and our use of its facilities. Therefore, the discussion is limited to the general principles closely related to our application, and the details are kept to a minimum.

The DC has a hierarchical tree structure, with a capability of access control at every node of the tree. We have set up a node structure of FACSIMILE, SITE, USER where the SITE is the message system used, and USER is the person accessing the DC. Whenever a user attempts to access a node, or file, the Data Computer scans the privilege blocks of all the nodes along the specified path name to determine what privilege should be allowed. If certain sites desire to be particularly secure, they may set a site password at that point in the path. In our implementation password protection is given to the users at the user nodes only. Thus to access a file in USER1, a user whose mail box is ISI and whose password is SECRET would require the data language commands;

```
LOGIN FACSIMILE ('PASSWORD');
LOGIN %LOGIN.ISI.USER1 ('SECRET');
```

If extra security is desired at the site level, say at UCL with the password LONDON, then the second line in the above example becomes:

```
LOGIN %LOGIN.UCL('LONDON').USER1('SECRET');
```

However, restricted access by means of passwords is only a part of the picture. For instance a user who has been granted the LOGIN privilege to the UCL node may well be denied say, READing, WRITEing, or APPENDing to the files contained in USER1. Therefore this selective access control mechanism can allow others to READ his messages but not to WRITE into them, if he so desires. Further details of this topic may be found in Reference 13.

The file FAX, where all the facsimile messages are kept, contains a list of message items called MSG1, MSG2 etc. which are formatted versions of the facsimile messages sent to the Data Computer. Each message space is divided into two areas, of these, USERLIST contains a list of vector names representing the users to whom the message was sent (e.g. UCL.PTK , BBN.SY ..), and the second, FAXDATA contains the actual facsimile message. The FAXDATA space is further divided into two areas. The HEADER contains the reproduction parameters, and BINDATA contains the binary data associated with the message.

Every user of the Facsimile System is assigned a private message vector, and each message has associated with it its status (Active, Read, Deleted etc.). The file descriptor for each message contains a user list of its addressees. Whenever a message is retrieved, the appropriate entry in the user list is marked. Thus a positive indication of message retrieval is obtained for each addressee. The USERLIST provides a link between the user vectors and the message pool, FAX, and it is systematically updated for formalising messages to new addressees.

It should be noted that the use of the Data Computer and MSG system is reasonably complex. The DC is designed for computer access and provides both machine and human readable responses.

```
@H 31 <CR>                                  Connect to D-C

@R F S 203 <CR>                             Socket 2Ø3

@I C P  CR                                  ICP

Trying ...

Open

;ØØ31  771Ø15Ø92551    IONETI:   CONNECTED TO LONDON TIP 12ØØØØ
;J15Ø  771Ø15Ø92559    FCRUN:    V='OC-1/Ø1.13' J=1 DT='MONDAY',
                                 OCTOBER 15, 1977 Ø8:59:26-est
;0041  771Ø15Ø92933    ONCINX:   DATACOMPUTER GOING DOWN IN 6Ø MIN
;J2ØØ  771015093021    RHRUN:    READY FOR REQUESTS

LOGIN FACSIMILE ('PASSWORD'): <CR>          Connect to Facsimile Node

-----                                       D-L Messages

.I210 .....                                 Sync

OPEN FAXP: CONNECT FAXP 262146; <CR>        Prepares O/P Port

OPEN %%LOGIN.ISI.KIRSTEIN('SECRET').CCA%771Ø15Ø92551: <CR>

---

FAXP = CCA%771Ø15Ø92551; <CR>               1 Send Data

---

CLOSE CCA%771015092551: <CR>                User Tidies
---
DISCONNECT FAXP: CLOSE FAXP:  CR            File and Port

                                            User Closes Connections

Closed                                      Connection Closed
```

Figure 5 An Example of Data Computer Access

3.5 The XGP System

At several sites on ARPANET, there are XGP printers. The XGP is an experimental device built by XEROX (Reference 14). It is driven by a PDP-11 which itself is controlled by a TENEX. The XGP software system is itself complex. For historical reasons, only character orientated software is supported on the systems; the hardware could support a raster scan mode. There exist facilities on the TENEX both for accessing the Data Computer and for any desired character set. By defining a six bit character set of rows of dots (the binary patterns 0-63), and by using the other seven bit combinations for control functions, a software system has been built which will print facsimile files on the XGP.

To describe this software in detail is irrelevant to this paper and not fruitful. The combination of hardware and software used derived only from that which existed already; it clearly has no relevance to an economically viable system. This exercise does show, however, that data input from one analog facsimile device could be output on a quite different type of device after staging on the Data Computer. Sample input from the Plessey KD-111 and output from the XGP are shown in Figure 6.

4. A COMMERCIALLY VIABLE SYSTEM

4.1 Introduction

The economics of the operational system outline in Section 3 require an analysis of the message processing, data storage, (i.e. the DC), and terminal construction costs. Of these the first two are in extensive use, and economically viable facilities. However, the facsimile terminal and the network access components of the system are experimental devices which require further improvements. Here we describe the principal components of a commercially viable system, and put forward some guidelines for its construction.

In Section 4.2 we outline the development required in the facsimile terminal. Sections 4.3 and 4.4 discuss the various aspects of Data Networks, Message Processing, and Data Base Management Systems.

4.2 The Terminal

It is clearly very desirable in this application to use a terminal with a digital interface at as early a stage as possible. Because the best storage is the original paper, it is important to chosse mechanisms which can be stopped on each scan line. This allows interruption of scanning or printing to compensate for the delays in the communication network. It may still be desirable in some applications, to have a local storage medium like floppy-disk or tape. This allows daytime loading of pages to be transmitted. In many cases the time to scan or print a document may be considerably faster than the transmission time; local storage could compensate for this transmission. However, in general the use of a local backing store would become an unnecessary luxury.

One problem which is avoided easily in the digital devices is the initial synchronisation. This takes 15 seconds in the KD-111, and can be replaced by a simple start/stop clock pulse in a digital machine. It is very important to have a good status information exchanged in terminal to terminal application. But in the form of usage envisaged in Sections 2 and 3, this information is not required. The uP only requires to have good status information and control about its local terminals. In unattended transmission

or retrieval, the human interface of Section 3.3 could be replaced by a command file. In this case good terminal recovery features must again be built into the uP control.

Data compression in the environment envisaged can use quite different algorithms from the more usual terminal to terminal case. Since the data transmission between controller and ADMS machine will use a standard data network, data transmission can be assumed perfect. This would allow use of algorithms, in which undetected errors make the data indecipherable. In practice, algorithms which ensure a good end-to-end transmission would be unnecessary.

The UCL approach of using one for all functions leads to unnecessary complexity in its programme. Clearly both data compression and decompression will use dedicated processors: the same one perhaps in half-duplex devices but different ones in full duplex. These same uPs may carry out encryption/decryption in some applications. The data compression/encryption/communication/sum check functions will probably be best carried out in one uP; they are serial operations; if necessary they can use several uPs, since in general they will use special chips (e.g. for encryption and CRC checking). These are high data rate function, which will require bit slice uPs in high performance (48K bps) terminals; in 8 and 16 bit uPs. The control functions again should be uP controlled. It is still an unresolved question how the uP software would be maintained. The terminals can remain simple, they can use Network Access Machine (NAM) techniques, and the NAM keeps up with changes in the Message and Data Management systems. Alternatively, known terminal types can be supported; the software for application development can then be developed centrally and the usual software releases made to the terminals. It is usually a complex process to update the applications programs and the intelligence of different terminals in a foolproof manner.

4.3 The Network

Very considerable effort had to be devoted to the communication aspects of the UCL development. This effort was due to the poor flow control in the terminal ports of ARPANET TIP. In the future at least with PTT networks, we may expect to use X25 packet interfaces and X25 networks in this application. This will have the advantage of the existence of standard communication packages. On the whole the simple character terminal procedures (for control) and the bulk data transfer procedures (for the binary data) envisaged for these networks will be ideal for the data transmission. Only simplified versions of X25 may be required; thus only two simultaneous calls in a half duplex terminal or four in a full duplex one, are required in the application of Section 3.

The staging of data on ADMS is an important technique in overcoming the problems of potential mismatch of speed between sending and receiving terminals operating at their maximum speed over a data network, With a normal current switched data network and direct terminal to terminal operation, different speed working is difficult (with complex multiplexing). In this mode of working there is no problem in the terminals having different speeds or data formats - though the latter may require some dynamic data transformation.

4.4 The Message Processing and the Data Base Management

In general the facilities provided by MSG (Reference 7) are quite adequate; by contrast for some applications the method of storage is not adequate. Some of the commercial message processing systems store also the textual data in one copy and manipulate header information by user lists. The addressees are merely sent notification of the availability of the text message. This technique, which is the one we are adopting for the facsimile and text transmission, works well when documents and messages are long - or have many addressees. A complication does exist when there are many connected message systems; then the only solution is to keep one copy of each document at each message system, and to manipulate only local user information. The alternative is to request the document each time it is required

across the systems, - this may cost more in processing charges than it saves in storage of multiple copies. It would be helpful if the responses and headers of the messages also had a machine readable portion. The fixed format of the messages does not make this change essential, but extension to internetworking with completely new message systems would make it desirable.

The facilities provided by the Data Computer are completely adequate for the Archival Data Base Management (ADMS) nature of the information and storage retrieval. For this type of application, where individual files will take minutes to transmit and output, the use of a ternary storage medium, like the Terabit store is clearly indicated. An attractive aspect of message systems with storage, is the ability to keep the confirmation on-line, so that it can be reused later. If only one copy of the data is shared amongst many users, it is debatable how charges should be divided amongst the users. They may like to require that the data be kept available even after they have accessed it once. The procedures provided for data deletion, which is related to charging philosophy, do require further study.

Since paper is such a suitable storage medium, there is a tradeoff between keeping the data on the Data Computer and in local paper form. Because modification of individual pages is impractical electronically there is less advantage in keeping it stored in electronic form. Here the tradeoff is between inputting the document and keeping it in storage for a long time.

Clearly the control aspect would be eased if the Message Processing System and the ADMS machine were more closely integrated. Mitigating against this is the need for a larger and cheaper storage machine for the facsimile data than for textual data (because of size and frequency of updates). The most probable development will be many more Text Message Processors than Terabit stores; the Text Message Processors will also be used for word and document processing activities, though the actual text composition will be done increasingly by local word processing systems (like the UCL FAX terminal!). It could be argued that the message

processing needed for the facsimile is so simple it could all be carried out on the ADMS. However, a key aspect of our philosophy is that the notification of the availability of facsimile documents is made on the message system the user customarily accesses. Thus we see the present system structure combining many intelligent FAX terminals, several message processing systems, and a few Archival Data Base Management Systems. The one substantial change which may be indicated is the provision of delayed delivery services. With present technology, facsimile requires in any case ½ - 4 minutes/page; though the 5 second/page service could be made available over 48K bps lines. Thus substantial documents would require non-negligible waiting lines. The Terabit store is a cheap storage machine, but has a very limited number of access ports. Consequently widespread use of the store for many short urgent messages would lead to access rate limitations. A facility for delayed delivery with perhaps overnight transmission, would allow the ADMS machine to rune more efficiently. For reasons to be discussed in the next section it would probably lead also to much lower communication charges. Finally, the delays incurred would probably be reasonable in many applications; where more urgent delivery is required higher charges might be imposed. It is particularly desirable to use a single technology, in which it is possible to pay for faster delivery at higher cost; this is far more satisfactory than to have to use a different communication system for the higher performance.

5 THE ECONOMICS

In any service, it is important to break down its constituent elements, and to consider how they may be introduced. It is certainly possible to estimate end-user cost by assuming that the constituent parts are charged at current rates; such an estimate may be useless because of the assumptions on which contemporary charges are based. We will try to assess the sort of charges the user might be asked to bear, and argue how these should be reduced.

In the service described in Sections 2-4 there are the following elements:

(i)	The cost of local communications to a Data Network for the transmitter	(C_1)
(ii)	Transmission Communication Costs on the Data Network	(C_2)
(iii)	Delivery Communication Costs to the ADMS and MP	(C_3)
(iv)	Retrieval Communication Costs from the ADMS	(C_4)
(v)	Retrieval Communications Costs in the Data Network	(C_5)
(vi)	Delivery Communications Costs to the Retriever	(C_6)
(vii)	Transmitter Terminal Costs	(C_7)
(viii)	Retrieval Terminal Costs	(C_8)
(ix)	Costs of Processing the Storage of the message and facsimile data	(C_9)
(x)	Costs of Processing the retrieval of the message and facsimile data	(C_{10})
(xi)	Costs of Storage of the message and facsimile data	(C_{11})

To give absolute or even approximate value to these costs is completely impractical; too many assumptions and parameters are involved. It is practical, however, to compare the costs of our proposed methods of facsimile usage with other methods of facsimile usage with other methods of message and facsimile

processing; in particular it is possible to assess the effect of different parameters.

5.1 Communication Costs

To reduce the number of parameters, we will assume that the constituent communication costs of the transmission to each addressee are the same (i.e. $C_1=C_6$, $C_2=C_5$, $C_3=C_4$). The communication cost of storing and retrieving a document is $(C_1+C_2+C_3)$ say C_c. It would be interesting to provide cost comparissons for C_c on public facilities. This figure is, however, critically dependent on volume of traffic and type of communication medium. The figures for C_c are discussed in Ref. 16; one example is given in Section 5.4.

If we had single addressees the cost of our method would be $2C_c$; if there were n addressees it would be $(n+1)C_c$. For the case of "one-to-many" transmission the extra communication costs C_3 and C_4 are offset as the number of addressees increases. Therefore the use of centralised storage (e.g. ADMS) becomes progressively more attractive.

If the real costs to a carrier of providing the service are analysed, the cost of the local access (C_1 amd C_3) will usually be much higher per message than the other components; the charges to the user are usually heavily influenced by the long distance communication; $(n+1)C_2$ are the main charges to the user with contemporary telephone charges for trunk distances. Over the new data networks, these distance charges are usually a much smaller proportion of the total cost to the user.

The cost to the carrier of local access to the ADMS and MP, i.e. $(n+1)C_3$ is comparatively low. If these two (or one) systems are heavily used, they will presumably have a heavy traffic, high speed, connecting pipe. Thus the real cost to the carrier will usually be dominated by the access costs to the terminal $(n+1)C_1$. In the normal method of multiaddress facsimile usage, the message is sent n times; thus the local

---access costs would be $2n\ C_1$ - substantially greater than their component in the store and forward case.

The above arguments apply to all message processing. Multi-destination message service tariff structures are already strongly influenced by these considerations - though the communications cost element is weakly separated out in the tariff. From the above we may draw the following conclusions:

(a) For multiple addressees with present tariff structures the communication costs to n addressees would be increased by (1+1/n)

(b) Assuming local access costs to the terminal are really the dominant item, the real costs are reduced by $\frac{1}{2}(1+1/n)$.

If the usage became popular enough for a carrier to provide the service, he would benefit from (b) and could pass to the end customer. Alternatively, an ADMS operator could be given volume discount on both received and transmitted messages, to reflect the reduced cost to the carrier.

The above arguments apply both to message and facsimile services. In our type of service, there is both a notification message and facsimile data. Since the notification message is short (typically 1 packet), its cost can be discounted. The tariff structure proposed for the new data services are largely based on volume of data (recognising the small real contribution of C_2). The tariff level is usually based on some comparison with Telex charges, though the relationship may be weak.

TRANSPAC (Reference 17) has proposed volume reductions of 40% for volume of data; some of the EURONET proposals (not from the PTTs) have been for 10-20% reductions. These proposals have been designed more from obtaining total revenue than marginal costing. Time of day volume reductions in TRANSPAC vary between 40% and 80%; the same charges proposed on EURONET were 50%. The TRANSPAC tariffs are presumably designed to generate usage; the comparitively low EURONET proposed reductions reflect their view that the network is intended for interactive bibliographic searches. In view of the fact that Facsimile traffic is 10-100 times less condensed than alphanumeric, it is essential to provide some large scale volume discounts, if this traffic should flow over the new data networks.

It is presumably, possible to ensure that this discount is only available to Facsimile traffic; one method would be to provide specific access ports (using leased lines or different frequencies) which qualify for the special discount. Unfortunately this approach is technologically unattractive; it makes the concept of modular integrated word processing, data or facsimile terminals economically unattractive. Present tariff structure (Reference 16) without discounts, indicate that the new data networks give no advantage for Facsimile traffic at peak periods over the use of public switched telephone networks; at off-peak periods they are even more unattractive. This situation would be radically changed if large scale reductions were provided. Here our proposed store and forward facsimile might be particularly interesting. If specific ADMS were licensed only for facsimile traffic, preferential volume tariffs could be levied on traffic between terminals and those machines. How the terms of license could be enforced, is not clear to the authors! These considerations may make the methods proposed in this paper much more attractive in closed communities, where marginal costs may be reflected in tariffs, than in Public Data Netoworks.

5.2 Terminal Costs

While it is indisputable that the UCL proposal assumes a fairly sophisticated facsimile terminal, typical sales costs for digital Facsimile devices are of the order of $10K. The intelligence of a terminal meeting the requirements of Sections 2-4 for synchronising and controlling data transfer could be provided by a Microprocessor. In our implementation we have used an Intel 8080 with 16K bytes of memory, and it is estimated that the extra intelligence would not increase terminal costs by more than about 20%.

5.3 Storage and Retrieval Costs

The cost of storage and retrieval of information is dominated by the amount of processing required in directory information. It is relatively insensitive to length of Facsimile data. Thus the MP and the ADMS are on separate systems, a first approximation is that these Figures i.e. C_9 and C_{10}, are twice the relevant text processing costs (because there are two links, one to MP and one to ADMS). The cost of this type of processing is of the order of $0.8/message. Clearly with long documents , the length of document must play a more significant part (on a TENEX, $0.3/1K characters).

The whole question of tariffs in MP and ADMS is complex; clearly the Terabit store is a cheaper technology/bit. Semi-commercial tariffs for the use of both TENEX with message facilities and the CCA Data Machine include components for opening connection, network I/O, File Space Storage, Node Space Storage, Ternary Storage, Connect time and CPU processor time. The procedures of Section 3 reduce any connect times to a minimum and reduce processing costs, because all composition is done in the terminal.

The question of costs and tariffs in message systems have been discussed in some detail by Panko.

The cost break downs for facsimile pages of 200K bits on the Data Machine are the following:-

Opening a Connection	$0.2
ARPANET I/O	$0.06/page
File Space	$0.0006/page day
Node Space	$0.001/day
File I/O	$0.002/page
Connect Time	$0.03/page
Processor Time	$0.4/page

Table 1 Typical Tariffs on the Data Machine

Many of these items are strongly application dependent. For example the node space depends on the complexity of the directory and tree structure of Section 3. The connect time assumed a one minute facsimile machine at 2.4K bps. These figures indicate, however, that the transmission processing costs are of the order of $0.8 for the message processing portion total, and 0.8 for the Data Machine portion. Retrieval processing costs are even harder to quantify. The message costs will be low; the cost is about $0.80 per access to the message processor, but many messages can be retrieved for this cost, and it depends on how many messages are stored. Thus the cost per message depends on the volume of messages retrieved at a time, and the complexity of the retrievers utilisation of message services.

5.4 A Simple Example

As an example, let us assume we are sending a facsimile message to ten addresses in the UK using EPSS (Reference 16) by straightforward terminal-terminal and terminal - MP/ADMS - terminal techniques. The sort of breakdown of costs could be as shown in Table 2, which are computed for different traffic loadings in Table 3.

A wide range of conclusions supporting the statements already made in this section can be drawn from these figures. It must be noted what we are comparing. The columns marked "Direct", include only the cost of transmission. Those marked "ADMS", can, because they use the Data Machine, also do a wide variety of personal filing at marginal further cost. The columns marked "direct", describe situations in which the called terminals may be busy; the ADMS ones do not have this danger, because the retriever determines when the document is fetched.

	2 Terminals and (1 Addressee)		11 Terminals and (10 Addressees)	
	Direct	With ADMS	Direct	With ADMS
Terminal ($/day)	22	22	120	120
EPSS (Fixed Costs) $/day	16	16	90	90
Communications ($/page/transaction)	0.18	0.18	1.8	0.18
Message Processing ($/page)	-	0.8	-	0.8
Message Retrieval ($/transaction)	-	0	-	0
Facsimile Processing ($/page/transaction	-	0.8	-	0.8
Labour Costs ($/page)	0.05	0.05	0.05	0.05

Table 2 Cost Breakdown of Different Uses at Store and Forward Versus Direct Facsimile Via EPSS

Loading (pages/day)	2 Terminals (1 Addressee)		11 Terminals (10 Addressees)			
	Direct	With ADMS	Direct	With ADMS		
				No of Retrievers		
				1	3	10
0	38	38	210	210	210	210
10	40	57	229	239	258	326
50	50	131	303	350	449	792
200	84	408	580	772	1164	2540

Table 3 Cost of Different Levels of Facsimile Transmission ($/Day)

At very low traffic levels, clearly ADMS makes no difference - the fixed costs predominate. At very high traffic rates, the ADMS costs are high; this is primarily because of the cost of the facsimile, and somewhat less ,the message, processing. The asymptotic increase factor caused by the ADMS is 9 for the one addressee case, and 6 for the ten addressee. This could be reduced drastically by a cheaper facsimile processing cost - which would be readily achievable with a software structure optimised for this purpose. The communication cost for the ten addressee case with ADMS is already only marginally (10%) above the case without. In fact with multiple addressees, the figures in Table 2 are too pessimistic with ADMS. If notification of a facsimile page, with some indications of content, is sent to multiple addressees, many will not need to retrieve the page, without ADMS, they may insist on receiving a copy. Since the dominant element of cost is the facsimile processing, which is directly proportional to the number of recipients retrieving their paper, the reductions which could exist are illustrated also in Table 3.

The main purpose of these figures is to point out what costs would arise having present technology and communication costs. The system with costs at Tables 2 and 3 could be put together tomorrow. The figures also show where the costs would lie, and therefore what technological and tariff advances must be made. Incidentally, in Reference 16 we have already shown that the tariff structure and levels of EPSS were basically too high for facsimile transmission. Only at levels of about 50 pages/day should it be considered at all. By attacking the communication costs and facsimile retrieval processing costs, the ADMS approach can be made much more attractive. The reasons why the first are considered artificial have been mentioned in Section 5.1. The reduction in the second will come from better Content Addressed Processor Designs, and better optimised software. Finally, for a small number of addressees, the message processing cost is significant. The cost can be almost removed by integrating the MP and ADMS.

5. CONCLUSIONS

We conclude that the type of storage and retrieval facilities outlined in this paper are a promising method of using facsimile equipment. The method is widely applicable, and would lead to significantly higher potential usage of such terminals. The method allows all the sophisticated data management techniques developed for text messages to be applied to facsimile files.

The extra complexity of the terminal is not a serious drawback. However, a significant capital investment in both Message Processor and Archival Data Management System is required.

Communication costs could be at worst doubled, and at best halved by the use of this technique. Key elements in reducing communication costs are bulk tariff reductions, and special tariff treatment of the ADMS. A key element in reducing processing costs is the reduction in retrieval costs by more optimised retrieval hardware. Archival storage costs are not a serious item, provided a technology such as the Terabit store is used.

ACKNOWLEDGEMENTS

We acknowledge the help of the British Library Research and Development Department, under Grant SI-G-172, who supported this work. It would not have been possible without the support of the Ministry of Defence (Contract AT/2047/064), the Science Research Council (Grant B/RG/67022), and the US ARPA, under the Office of Naval Research (N00014-77-G-0005), who made the use of ARPA Network facilities possible. Finally, many colleagues at the Computer Corporation of America, the Information Sciences Institute, and the UCL INDRA group, who helped in the work.

REFERENCES

1. Roberts L.G. et al, "The ARPA Computer Network", Comp. Comm. Netw., Prentice Hall, pp485-500, 1973

2. Kirstein P.T. and Wilbur S.R., "The Impact of Integrated Message Processing Facilities on Administration Procedures and Inter-Personal Interactions", Proc. Eur. Conf. Comp., Comm. Networks, pp 395-414, 1975

3. ---, On Time, "The Message Switching System with a Future", Tymnet Inc., 1977

4. ---, "HP 2026 - Data Entry/Communication System Reference Manual", Hewlett Packard, 1977

5. Spitzer F, "Typesetting in Word Processing Environment", Sicob 1977, Paris, pp 204-209, 1977

6. Newcomer J., et al, "An Introduction to XOFF", Carnegie Mellon University, 1973

7. Vittal J, "MSG Manual", University of Southern California, Information Sciences Institute, 1975

8. ---, "NLS Journal System User Guide", SRI, 1973

9. ---, "ARPA Network, Current Network in Protocols", NIC 7104, SRI International, 1973

10. Chandler A.S, "Network Independent High Level Protocols, Proc. Eur. Conf. Comm. Networks, pp 583-602, 1975

11. Kent S, "The Automation of Network Access Procedures by Means of a High Level Control Language", Ph D Thesis, University of London, 1977

12. Kirstein, P.T., "University College London ARPANET Project, Annual Report" 1 January 1976 - 31 December 1976, University College TR 34, 1977

13. Yilmaz, S. and Kirstein, P.T., "UCL Experiments in Facsimile Transmission Using Data Base Management Facilities on ARPANET", Proc. Eur. Conf. Comp. Comm. Network, 1978

14. Vittal, J. and Currier, "The ISI-XGP System", University of Southern California, Information Sciences Institute, 1975

15. Panko, R., "The Outlook for Computer Mail", 1976

16. Kirstein, P.T., "Choice of Data Communication Media for Transmission of Facsimile Information", Computer Netw. (in press)

17. Danet, A. et al, "The French Public Packet Switching Service; the TRANSPAC Network", Proc. 3rd ICCC, Toronto, pp 251-260, 1976

VIII THE USAGE OF THE UCL ARPANET LINK

8.1 Introduction

The UCL activity with ARPANET has a research aspect, discussed in Chapters 2-7 and a service aspect. In the service activity, we provide support to UK research groups requiring to access US resources attached to ARPANET. We also provide help to US groups using the UK resources accessible via the UCL ARPANET node. Finally, we provide the technical facilities to allow the access to take place.

The service facilities have improved, but not changed dramatically. The diagram of the UCL configuration, Fig 2.1 of Kirstein 1977, is still valid. The usage of both the Culham System 4/72 of the ULCC CDC 6000/7000 is still strictly experimental. Service has started in a limited way via EPSS. This will be increased progressively during 1978. The RL IBM 360/195 and the RSRE GEC 4080 have remained attached via the PDP 9A.

We have made consistent measurements of all traffic passing through the UCL PDP 9A. This year, for the first time, we have analysed this usage. The resulting usage figures are discussed in Section 8.2.

There have been no dramatic changes in the types of usage. A general discussion of usage patterns is given in Section 8.3; the activities of the higher activity groups are discussed in Section 8.4.

8.2 Objective Figures of Usage of RL IBM 360/195

In the earlier years, we had developed a program called QUES to analyse the usage of ARPANET via the PSTN ports of the TIP. This activity required the dedicated availability of one PDP 9; the PDP 9s were so heavily loaded during 1977, that almost no use of QUES was made in this period. The only exception was a one week period, in which QUES was used on a 24 hour / day basis; the data acquired was then correlated with PO measurements on line usage. These measurements, discussed in STOKES 1977B, showed good correlation between QUES and PO measurements.

During 1974-1977 we have consistently noted all usage of the UCL ARPANET node through the UCL PDP 9A. This usage includes both that of Hosts on ARPANET accessing the RL IBM 360/195 or the RSRE GEC 4080, and of UK users of the RL 360/195 accessing ARPANET facilities. The usage data collected was refined as the PDP 9 software was improved, and no consistent analysis was attempted until 1977.

During 1977, the 1976 and early 1977 data on PDP 9A usage was analysed. A detailed discussion of the results is given in STOKES 1977A and STOKES 1977C. Two periods were chosen for the analysis; all of 1976 and the first half of 1977. The actual usage (in connect time) of the RL IBM 360/195 from different ARPANET sites for the two periods is shown in Figs 8.1. From the identifying information furnished by the users, it was often possible to determine the scientific community from which the usage originated. The relevant figures are shown in Fig 8.2. Finally, the duration of file transfer could be measured separately. Here the sites involved in the file transfer are indicated in Figs 8.3. In each of these curves the area is proportional to the average usage during the period.

FIGURE 8.1a: USAGE OF RL 360/195 FROM VARIOUS ARPANET HOSTS FOR 1976

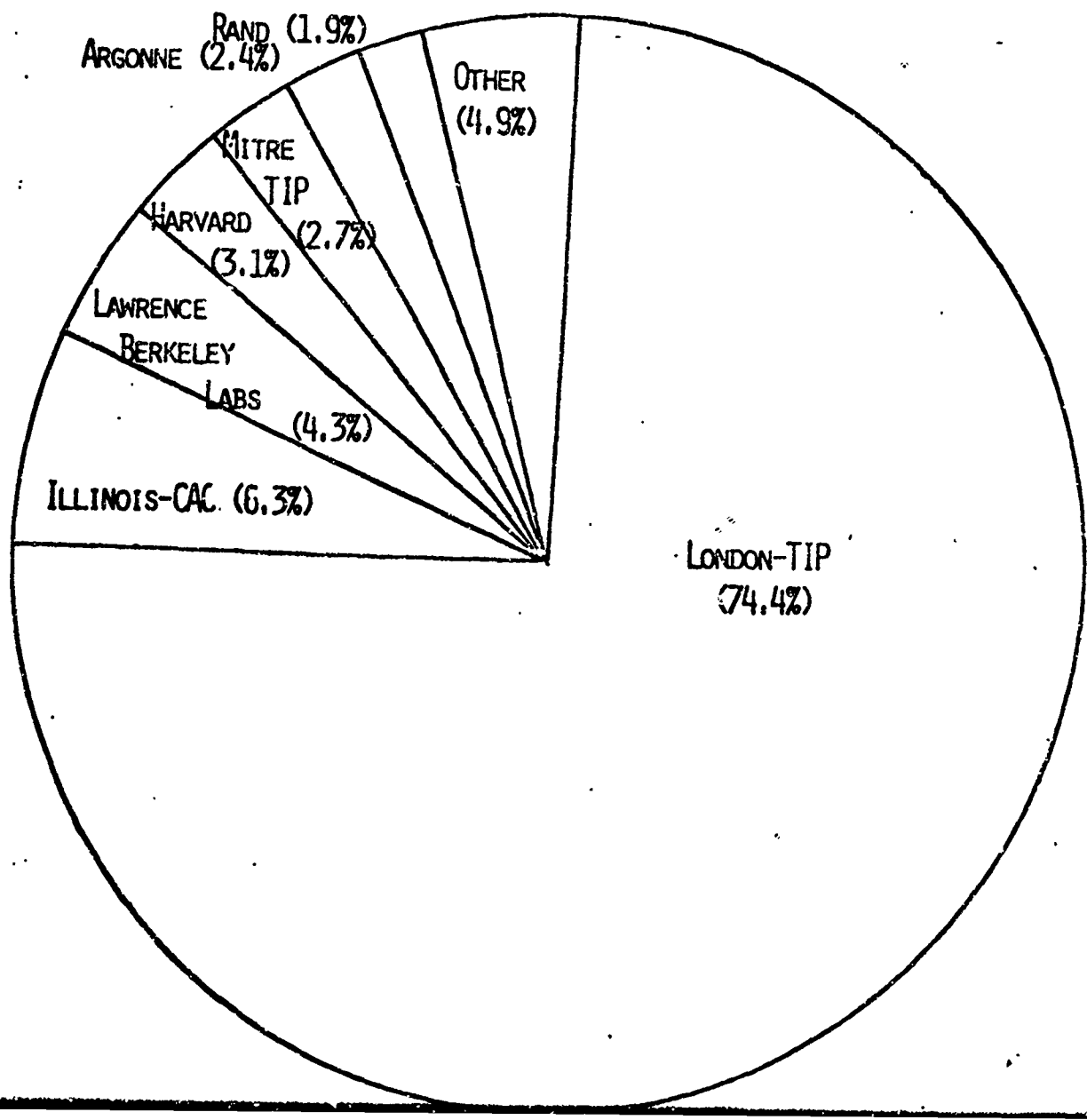


FIGURE 1b: USAGE OF RL 360/195 FROM VARIOUS ARPANET HOSTS FOR THE FIRST HALF OF 1977

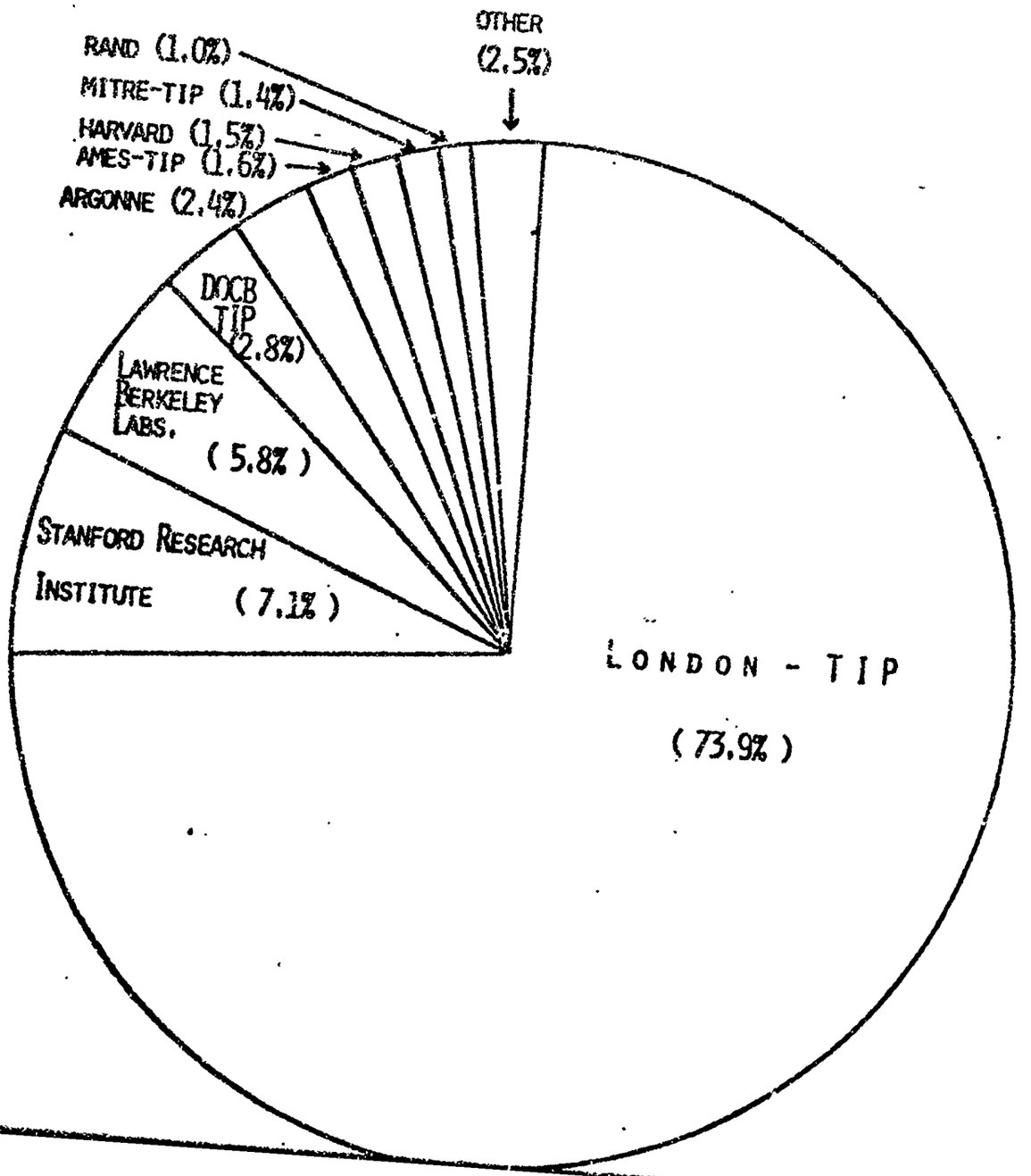


FIGURE 8.2a: USAGE OF RL 360/195 BY VARIOUS RESEARCH PROJECTS FOR 1976

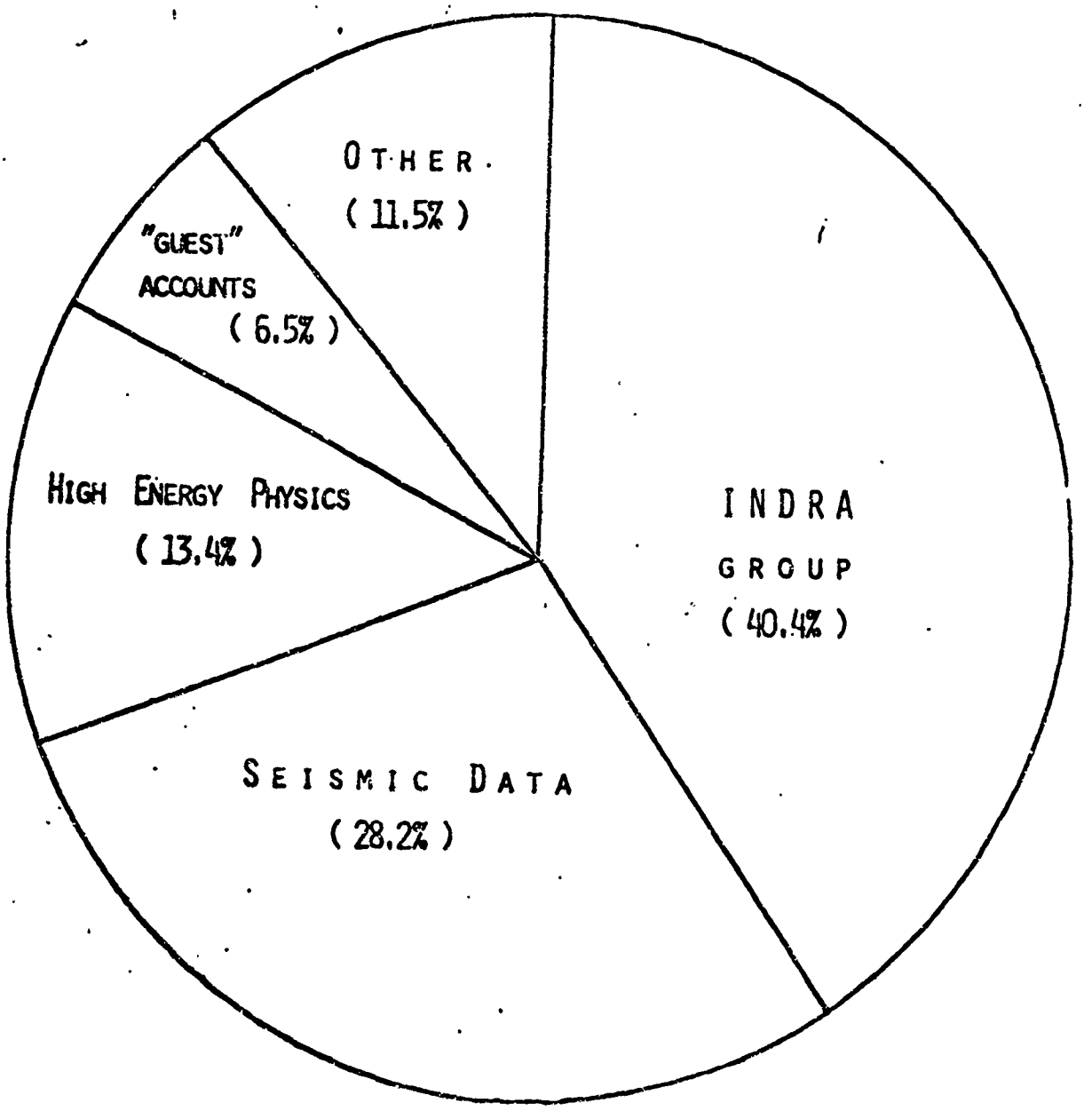


FIGURE 8.2b: USAGE OF RL 360/195 BY VARIOUS RESEARCH GROUPS FOR THE FIRST HALF OF 1977

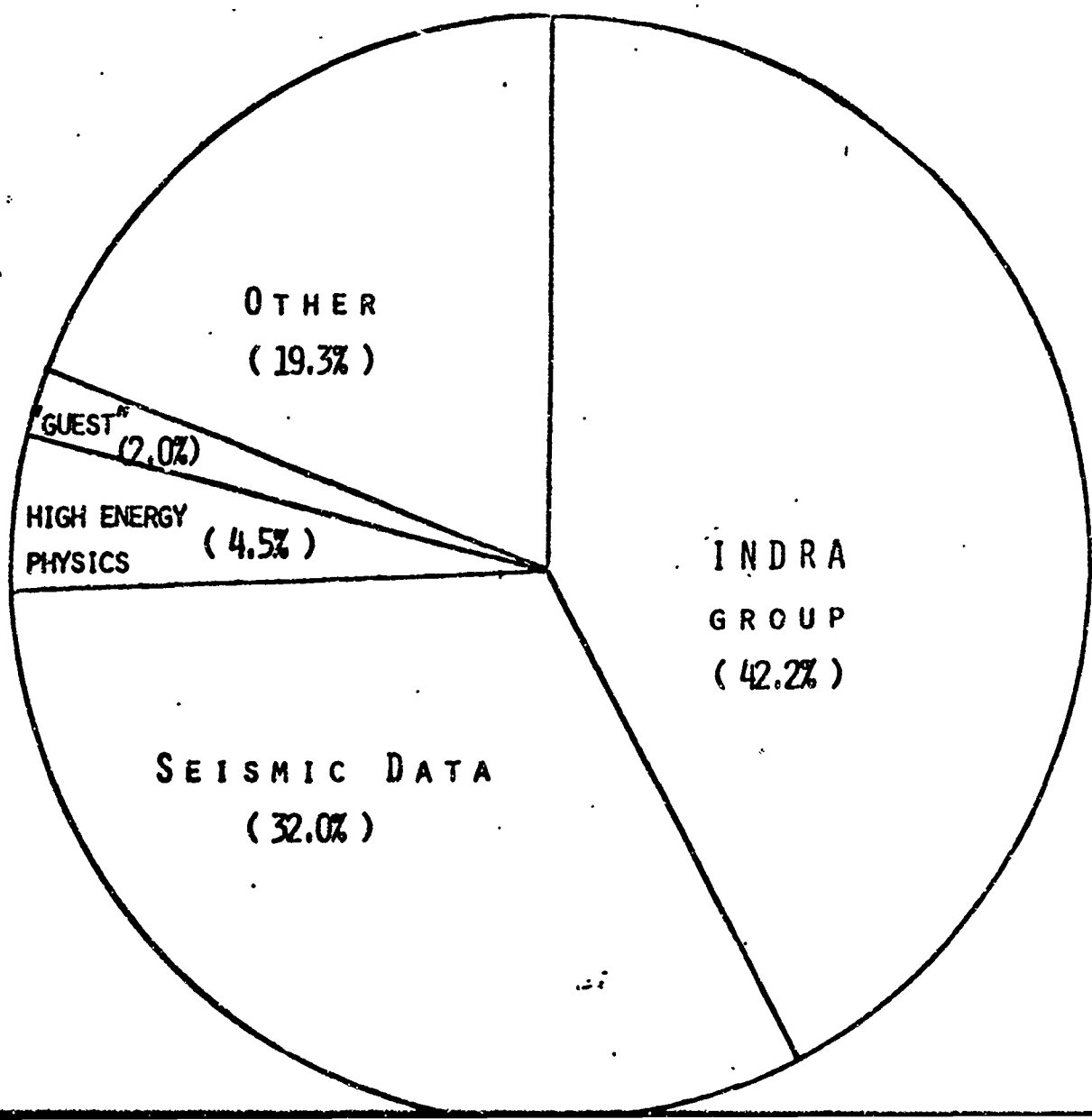


FIGURE 8.3a: USAGE OF ARPANET HOSTS FOR FILE TRANSFERS FOR 1976

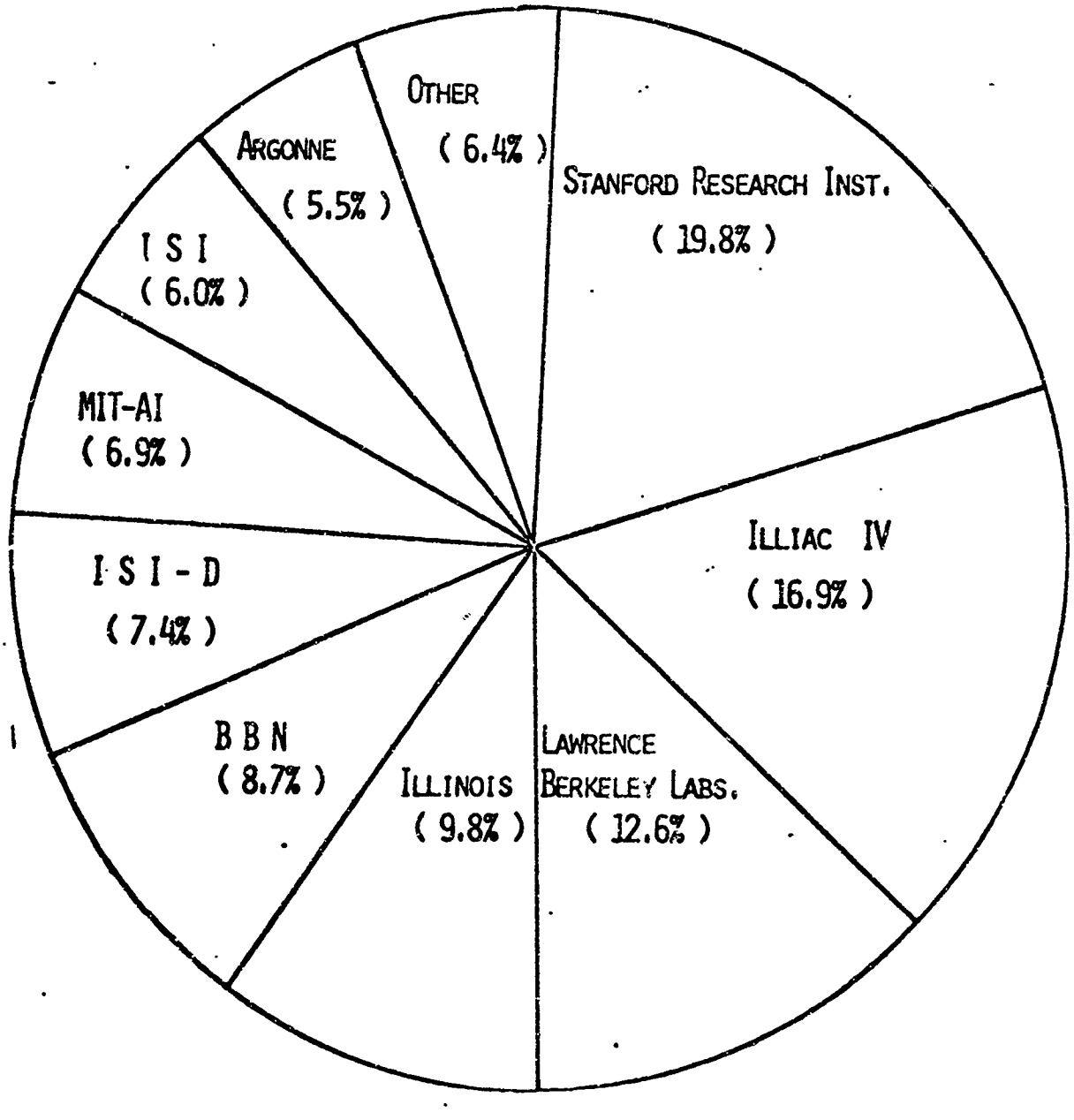
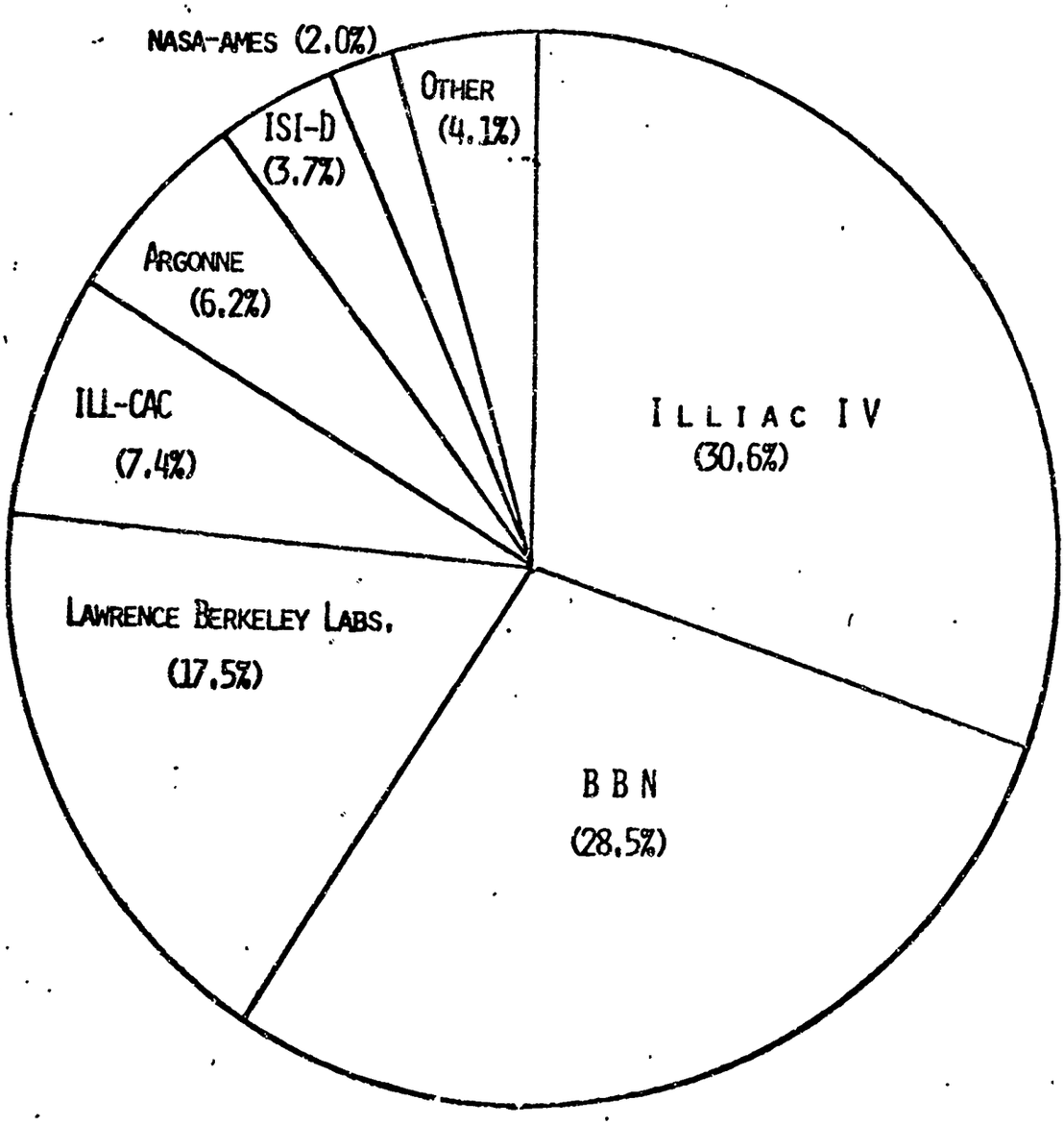


FIGURE 8.3b: FILE TRANSFERS PERFORMED BY ARPANET HOST FOR THE FIRST HALF OF 1977



8.3 General Characteristics of Usage of UK Research Group

During 1977, a number of the inactive users of ARPANET had their permission to use the network withdrawn. In addition, one very active user, the British Library usage of NLM, was transferred from ARPANET to the PO Data Base Service, the experimental basis on which they had used ARPANET terminated with the experimental project STEIN (HOLMES 1977). The UK research groups approved for use of ARPANET at the end of 1977 are listed in Table 8.1. Of these 42 (34) groups 13 (10) may be classified as "highly active", while 18 (12) have low activity status; the figures in parentheses represent the corresponding 1976 figures.

There have been no significant changes in the mode of access or types of facility connected to ARPANET or to the UCL node during 1977. There has been some experimental access via EPSS, the UK Experimental Packet Switched Service, during the year, but the system was not made available by the UCL for a significantly long period in the day for a pattern of usage to develop. The EPSS accessibility will be increased considerably during 1978, and an attempt will be made to transfer more people over to that mode of access.

Some of the research groups are making increasingly sophisticated use of the network. These groups include Blacknest Research Establishment, Cambridge U, Durham U and Salford U. The INDRA group of London are, of course, particularly active, and their use has become very sophisticated during 1977. (see Chapters 3,4 and 5).

8.4 The Most Active Research Projects

Some of the more active research projects are discussed in more detail in this section. The discussion is based largely on reports furnished by the research groups themselves.

Table 8.1

UK RESEARCH GROUPS WHO WERE

APPROVED ARPANET USERS DURING 1977

Organisation	Name	Project	Site
Blacknest Research Estab.	F. Grover	Seismic Data Exchange	ISI, RL
Bristol University	Dr. J. Alcock	High Energy Physics	LBL, CMU
Brunel University	Dr. L.D. Phillips	Perception of Probability	ISI
Cambridge University Computer Laboratory	Dr. A.C. Norman	Algebraic Manipulation Systems	ISI
Communication Studies	Dr. R. Pye	Teleconferencing	OFFICE-1
Culham Laboratory	R. Endsor	ALGOL Generator	ILLIAC IV, BBN
Durham University	Dr. F.D. Gault	Exchange of High Energy Data & Software development	LBL
Edinburgh University	M. Gordon	Proof Generating System-LCF	SU-AI
Edinburgh University	Dr. R. Burstall	Program correctness	ISI, UCLA
Edinburgh University	Prof. D. Michie	Evaluation of Heuristic Programs	ILLIAC IV SUMEX AIM
Essex University	Dr. J.M. Brady	Visual Information Processing	MIT-AI
Hatfield Polytechnic	Dr. G.M. Bull	Basic	NBS
Hatfield Polytechnic	Dr. A.V. Stokes	User Interfaces on Heterogeneous Computer Networks	ISI
IEA Coal Research	Dr. K. Gregory	Teleconferencing	LBL
ICI Ltd.	J. Barnes	DOD real time language evaluation	ISI
Liverpool University	P. Leng	ALGOL 68 development	CMU
Manchester University	Dr. R.N. Ibbett	MU5 modelling	CMU
Medical Research Council	R.T. Wilkinson	Teleconferencing for neurophysiology collaboration	BBN
Ministry of Defence	D. Curry	Collaboration with US Army Material Command HQ	OFFICE-1 ISI

Table 8.1 (contd)

National Physical Laboratory	Mrs. J.C. Armstrong	Networks, Protocols	ISI, BBN
National Physical Laboratory	Dr. M. Overton	Minimization problems using "MACSYMA"	MIT
North London Polytechnic	Dr. A.P. Johnson	Synthesis design of organic molecules	Harvard
Open University	Dr. M. Eisenstadt	Computation models	ISI
Oxford University	S.J. Hague	Numerical Software	Argonne, BBN
Oxford University	Dr. T. Quirk	Nuclear Physics	Harvard, BBN ILLIAC IV
Oxford University	J. Moussouris	Tensor Algebra	MIT
Oxford University	Prof. C.A.R. Hoare	Language Design	SRI-KL
Post Office	C. Broomfield	SIMP Experiments	BBN
Queen Mary College	G. Coulouris	Internetworking	PARC, MIT
Reading University	Prof. R.W. Hockney	Parallel Computing	ILLIAC IV
Royal College of Art	Dr. P. Purcell	Computer Aided Design	Harvard, MIT CMU, UCLA
Royal Signals and Radar Establishment	N. Neve	Support of US evaluation of CORAL	US Navy Lab.
Royal Signals and Radar Establishment	Dr. J.M. Taylor	Networking	NELC MITRE, RADC
Rutherford Laboratory	Mrs. S. Ward		
Salford University	Dr. G. Laws	Supersonic Flow Problems	ILLIAC IV
Software Sciences Ltd.	D. Pearce	Evaluating CORAL for DOD	
Systems Research Ltd.	Dr. G. Pask	Contact with US Army	
Thames Polytechnic	T. Crowe	Networks Editors	SRI
UCH Medical School	Dr. L. Kohout	Evaluation of Medical Data	SUMEX AIM
University College London	Prof. P.T. Kirstein	Problems in Computer Network Design and Applications	
University College London	T. Westgate	Teleconferencing	
York University	Prof. I.C. Pyle	Conference Word Processing	ISI

Blacknest Research Establishment

The Blacknest Research Establishment's project to exchange seismic data with centres in the US via ARPANET began in quite a modest way in 1974, when the ARPANET link was first established at UCL. They began by accessing data from the Seismic Data Array Center (SDAC), Washington and then began to set up their own database on the RL 370/195 for US counterparts (Blamey,1976). These data are now being accessed by the Vela Seismological Center, SDAC, and the National Earthquake Information Service (NEIS). New partnership identifiers have been allotted to VELA and NEIS to ensure rapid access to the BDAC data base.

The NEIS is the foremost collector of earthquake data from diverse sources. Computers and computer networks are opening many opportunities for improving the quality of data which NEIS collects, and by participating in the transmission of data by computer networks, Blacknest is assisting in a project that will permit near-real time location of important earthquakes.

NEIS have always received data from Blacknest, but rather indirectly. By transmitting data via ARPANET, it may be obtained far more quickly than before. The current data files on the 370 are in a different format from that used by NEIS, so Blacknest have implemented software which reformats the data in the standard NEIS format, before sending it to the PDP-10 at the Information Sciences Institute, California (ISI). Furthermore, BRE is acting as an agency for the onward transmission of seismological data from the University of Leicester to NEIS via the ARPANET.

In the other direction, during 1977, Blacknest has started receiving daily bulletins from a station in N.W. Canada. These are received from Ottawa via the Vela Center, who make the data available in files on the ISI machine. In due course, when NEIS is fully integrated with ARPANET, they hope to get much more quickly the seismic epicentre location data which they currently receive via air mail. Blacknest state that their attachment to ARPANET greatly aids their research.

Bristol University

Alcock of the Physics Department, Bristol, has been using ARPANET via the dual RL 370/195 to collaborate with Kelly of the Particle Data Group at the Lawrence Berkeley Laboratory, California (LBL). They are engaged on a joint project to analyse pion-pion to nucleon-antinucleon scattering data. Use of the ARPANET has enabled them to initiate, modify and execute jobs on the CDC 7600 at LBL from Bristol, and on the 370/195 at RHEL from Berkeley The reason for using two sites is desirable because certain facilities available at Berkeley cannot easily be reproduced at RHEL. Moreover it is immensely useful for Kelly to scan the results of the 370 runs from LBL and to keep up-to-date log sheets on the status of the runs. Clearly, this work could not be contemplated without the network and the ability to transfer files even though their use of the TIP is not great. The slowness of file transmission, between the 7600 and 370, has proved a serious shortcoming. However, access to ARPANET is essential since the project is about half complete.

Cambridge University

Use of the ARPA network through the UCL-TIP has enabled the Cambridge symbolic algebra group to continue and extend existing close ties with the corresponding group at the University of Utah, Salt Lake City. In March, Hearn (from Utah) visited Cambridge to discuss the use of his algebra system "Reduce". He was provided with a copy of a symbolic integration program developed in Cambridge, which was designed to coordinate with Reduce. The major use made of the network has been tied to Hearn's work implementing this 8000-line program on his computer, Cambridge's responses to problems so far uncovered and joint work exploring the limitations of the algorithms on which the program is based.

Partly as a result of an ARPA contract, in September, Fitch left Cambridge to take up a one year visiting position in Salt Lake City. He is due to return in the autumn of 1978. In the meantime the remaining workers have been able to take advantage of the network to avoid Fitch's movement disrupting the main research in which all three had been involved.

The Cambridge group were given access to an account on a machine at DEC Marlborough towards the end of the year because Utah U. were negotiating for a DEC 20-40 to replace their PDP 10. The provision of this account, coupled with the presence of Fitch in Utah and active work at both ends on the integration program has meant that in the last few months Cambridge has made heavier use of the ARPA network than for some time. The integration work has also been substantially helped by access to MACSYMA at MIT.

The Cambridge University computing service hope, in the very near future, to provide access to the London TIP via EPSS rather than by use of dial-up lines. This should allow for remote access to the Cambridge computer and for file transfer mechanisms enormously more reliable than the ones currently used. Both of these changes, would make transatlantic cooperation smoother. Also, Cambridge hope to use ARPANET as an adjunct to the new research project in the field of computer algebra (currently the subject of an SRC grant proposal), where as well as their existing links with the University of Utah there may be the possibility of fruitful collaboration with a group in Hawaii.

Culham Laboratory

The Culham fusion laboratory has joint interests with US laboratories in the same field. Work has been going on this year to enable users of the Culham Modular One Satellite to access ARPANET via the UCL PDP-9 Gateway computer. The satellite acts as a focal point for computing services provided to Culham terminal users, and as a server for remote ARPA users. A file transfer protocol has been implemented allowing Satellite users to perform file transfers to and from remote ARPA hosts. Although usage of these facilities has to date been limited, the feasibility of providing full network access within the Satellite context has been demonstrated and it is hoped that trial projects will take place in the near future.

Durham University

Gault of the Physics Department, Durham U, heads the UK section of the International Particle Data Group. His work is concerned with the development of high energy elementary particle-scattering databases.

Since the last report Durham have continued their collaboration with the Particle Data Group at the Lawrence Berkeley Laboratory. The work has involved the development and use of the Berkeley Data Management System (BDMS) on the Rutherford Laboratory 370/195, and the development and use of the Particle Physics Data Language for encoding data.

During the past year the most up to date version of BDMS, V2.0, was implemented on the RL 195. This was the first implementation to depend completely on ARPANET facilities, in that the source program was transferred using FTP, rather than tape, and debugging of machine dependent code was facilitated by being able to work on both the 195 and the LBL CDC 76/6600 from Durham. The same observations apply to the output processor or report generator, written in Durham and now implemented at LBL. In brief, the systems side of Durham's project on both sides of the Atlantic benefited immeasurably from ARPANET service.

Apart from systems use, there was considerable reliance on ARPANET by the project's Data Base Managers, Roberts at the Rutherford Laboratory and Read in Durham. They used ARPANET both to transfer data to and from LBL, and to make comments on BDMS, as BDMS users.

As there is a growing community of BDMS users in the US, LBL has taken advantage of the PLANET conference facility available on the BBN-TENEXB. This allows users to request modifications, report bugs, or to comment on proposed modifications before they are implemented.

Because the project is supported by SRC(NG 0633-3) and uses the RL 195, Durham report annually to the Nuclear Physics Board, and to the Rutherford Laboratory Computer Advisory Committee. The relevant paragraph in the SRC report was:

> "ARPANET has grown in usefulness to the project as its file transfer procedures have improved and it will play an even greater part when our computer searchable data bases are available".

The most significant event in the past year for the project was the release of Durham's computer searchable data bases for use by the nuclear physics community. As these data bases are not yet duplicated at LBL, there may be a small increase in ARPANET activity from US users wishing to access data. This will pass in about a year's time when a duplicate base will be maintained at LBL.

Edinburgh University, Computer Science Department

A new proof-generation system for LCF has been designed and implemented in UCI-LISP. Gordon and Milner of the Computer Science Department had not use for ARPANET until Milner took a tape of LCF to the Stanford AI Laboratory (SAIL). Gordon monitored Milner's progress over the network.

Edinburgh University, Machine Intelligence Research Unit

Carhart of the Machine Intelligence Research Unit (MIRU) has been using the ARPANET to access the SUMEX-AIM computer facility at Stanford University, initially over the link from the Royal Observatory at Edinburgh and more recently via a direct dial-up facility provided to MIRU by the Science Research Council. His usage typically amounts to about six hours of connect time per week, He has used the facility in various activities related to the Heuristic Programming Project (HPP) at Stanford, which holds an ARPA contract.

These activities consist primarily of:

1) Assisting in the maintenance and development of two AI programs(applications programs in Chemistry), CONGEN and REACT, which have grown out of the DENDRAL project, a part of the HPP;

2) Running assorted test cases through CONGEN to verify the accuracy of a new, compact and very efficient version of CONGEN which Carhart developed on the Edinburgh PDP-10; and

3) Sending and receiving messages to the extent necessary to co-ordinate the CONGEN developments at the two sites.

The main usage of the ARPA Network from the Computer Science Department has been by Ibbett and Djordjevich, working with CMU-10A at Carnegie-Mellon University. The work is concerned with the Instruction Set Processor (ISP) description language, for which a compiler and simulator has been implemented at CMU. These facilities are not available elsewhere, and it is important to the project that contact be maintained with the staff at CMU who have created ISP descriptions of several computers. The work at Manchester has been to create an ISP description of MU5. The next stage of the project is to run a number of test programs through the simulator and compare the results obtained for MU5 with those obtained for other computers.

Work started on the MU5 ISP project in March, but the amount of network time used has varied considerably from week to week. A typical session has been of 3-4 hours duration giving an average usage of around 15 hours per week. The equipment in use at Manchester (a Creed Envoy Teleprinter linked through an acoustic coupler) has proved adequate for their needs, supplemented by the occasional line printer listing sent through normal channels from CMU. Attempts to use the EPSS link have been abandoned due to continuing operational difficulties.

Some occasional use of the ARPA Network has been made by Lindsey, initially working with CMU-10A, but mainly with CMU-CMMP which has only recently been made available to remote ARPA users. He has been investigating suspected faults in the Algol 68 compiler in conjunction with Hibbard of CMU.

Usage of the network during this period was wholly related to the project "Computer-Aided Synthetic Analysis" which Johnson of the Chemistry Department has been pursuing in collaboration with a group at Harvard University under the direction of Professor E.J. Corey. Work on this project may be divided into two categories:

a) Development of new program modules and extension of the data base both of which involve the creation and editing of alphanumeric files which can be done using a simple teletype terminal.

b) Testing the enhanced program and debugging new program modules. This can only be done via a graphics terminal since when the program is running all communication occurs via a digitizer tablet and c.r.t scope.

The SRC has awarded Johnson a grant for the purchase of the graphics terminal equipment. Delivery is expected in March 1978. Therefore, use of the ARPA network has been limited to those aspects of program development which have been described in (a). In this connection, although the network has been used very sparingly, it has proved extremely useful in allowing real collaboration between individuals whose expertise is in differing areas (computing science and organic chemistry in this case).

Specific program modules whose development has been facilitated by the ARPA network include that concerned with the "Quinone Diels-Alder Reaction" and the "Ring Expansion/Contraction" package. Work on these modules is near to completion and the results will be published in the near future,

Once the graphics terminal has been installed, Johnson expects to make much greater use of the network for program testing and for file transfer between Harvard and the SRC DEC-10 at Edinburgh (via the Rutherford 370/195).

Oxford University, Numerical Algorithms Group

The Numerical Algorithms group of Oxford Computing Laboratory, are collaborating with a group from the Computational Mathematics Division, Argonne National Laboratory (ANL). Their research includes the devleopment of new algorithms and sophisticated tools for transforming mathematical software. Their use of the ARPANET consists of interactive use of the Argonne TSO system, and file transfers between Argonne and Rutherford (whose ELECTRIC System can be accessed directly from a terminal in Oxford). Transfers sometimes involve using BBN as a 'staging post' because NAG's filestore limit at Rutherford is low. Using FTP has proved difficult because of the lack of documentation from ANL, and software bugs.

Oxford University, Nuclear Physics Department

Quirk of the Department of Nuclear Physics, Oxford, leads a group who are working in collaboration with groups at Harvard, Chicago and Illinois. This work concerns experiment 398 at the Fermi National Accelerator Laboratory, Illinois. Muon-scattering data gathered at Fermi is reduced and analysed using software implemented on the RL 370/195.

The collaboration has been working successfully for several years. The group has not been particularly active in the last six months as it has found it difficult to make and keep contact over the network for periods of more than about 30 minutes. It expects that this is due to bad telephone lines in the United States, and particularly bad when coming out of the network using local dial-up facilities in the Chicago area.

The group is continuing to make use of the network as a way of transmitting up-dated versions of programs. These can be transmitted by FTP, with little difficulty, to the University of Illinois. One of the groups collaborators at Fermilab has been making use of the network for job submission to the Rutherford 370/195. He has not attempted more than this since modifying programs remotely is difficult for the reason given above and secondly because the slow response time makes using the network difficult.

The group has a continued need for the link as their ability to swap programs quickly and easily enables them to be part of a wide-spread collaboration of Fermilab and the Universities of Harvard, Illinois and Oxford.

Queen Mary College (QMC)

The Computing Laboratory at QMC, under Coulouris, is engaged in research aimed at developing software and system concepts to enable low-cost information processing systems to be more effectively applied. The most important goals are the development of highly-responsive interactive systems, and the provision of effective and generalised communication and cooperation between distributed systems. Use of the ARPANET enables them to obtain information about research related to theirs in the US. This may be in the form of online text unavailable in printed form or in the use of systems developed as research projects.

Usage has been similar to last year's usage. That is, the ARPANET has been used, in the main, for retaining contact with other US Computer Science groups, and exchanging Unix software.

The main limitation on the computing laboratory's greater use of the network is the difficulty in transferring files from their local unix systems to the network.

Hockney's group in the Computer Science Department, Reading University is engaged in a project to evaluate and develop rapid elliptical solvers and particle/mesh algorithms for parallel architecture computers. During the last year they have continued to use ARPANET to access the Illiac IV (I4-TENEX) and the NASA/Ames IBM 360/67 (Ames-67) as part of the project to evaluate and develop algorithms for parallel architecture computers.

They have implemented the P^3M algorithm in the CFD language for the Illiac IV, and this is currently undergoing final testing. This has necessitated fairly heavy use of ARPANET to access Ames-67 where the CFD code is compiled, and I4-TENEX to submit it to the Illiac IV. Without ARPANET this work would not have been possible.

The PSTN telephone costs in accessing ARPANET through the the London TIP last year began to exceed the group's Datel budget, so connection via the Rutherford Laboratory 360/195 to which they have a leased line was investigated. Connection via Rutherford was somewhat less convenient than using the TIP

To develop a large code (like P^3M) the Reading group needed to be able to route line-printer formatted compiler listings across their network to our Rutherford work station printer. This use of ARPANET file transfers to Rutherford was not without problems.

Recently two other groups within the Department of Computer Science at Reading University have been attempting to set up collaborative projects with co-workers in the U.S.A. and have made exploratory use of ARPANET to investigate the feasibility of their plans. The group studying the evolution of galaxies have successfully accessed the MIT-MC host while the one working in molecular dynamics have encountered some problems in using the Argonne National Laboratory IBM 360/195.

The Reading group hope to continue their use of ARPANET to access both Ames-67 and the Illiac IV to implement other algorithms, and expect this to use a similar amount of connect time to the present (10-20 hours per week). The other two groups' usage is expected to be somewhat less in terms of TELNET connection time but with substantial use of File Transfer facilities, as their aim is to run production programs.

Royal Signals and Radar Establishment (Malvern)

RSRE (Malvern) use of the ARPA network falls under two categories. The first concerns the provision of a GEC 4080 computer to other network users. This computer was originally made available via the network to the US DOD so that they could evaluate the programming language CORAL 66. The subsequent decision by DOD to develop their own language has led to a falling off of the use of the 4080 by the DOD, but some continuing use has been made of the machine by UK based users.

The second use of the network by MOD is in connection with the development of a new programming language by DOD. The MOD are actively participating in this project in a number of areas. Extensive use of the network is made by DOD for management of the project, running of economic models, and the development of future plans with other participants. MOD intends to use the network in the future to monitor the project, and to run prototype compilers and support software developed by DOD, so that an assess-

Laws and Walkden of the Mathematics Department have been working on a project to develop and verify algorithms for predicting steady supersonic flow fields. Such algorithms involve very complex multi-dimensional matrix calculations that are impractical for ordinary serial processing and thus the ILLIAC IV parallel processor machine is being used for program development.

The period covered by the report is divided into two distinct parts: November to June and July to December. During the latter part, use of the network was minimal.

Between November 1976 and June 1977, an average weekly connect time of 268 minutes was used to access I4-Tenex, to test and run 3-D supersonic flow calculation programs on the Illiac IV computer. Their usage involved the following types of operation:

(i) transferring files between the 360/195 and I4-Tenex

(ii) file creation and editing; job submission; examination of numerical results and output files produced by the I4-Tenex system.

Translators installed on the Rutherford Laboratory 370/195 computer produce programs in forms suitable for running on the Illiac IV. computer. Consequently, a substantial portion of access time to I4-Tenex has been spent transferring program files from the 370/195.

Two complete sets of program files were transferred, tested and subsequently run on Illiac IV. The first during December /January and the second between March and May. Improved FTP software has meant that the number of file transfers that had to be abandoned because of the length of time involved, has been significantly reduced. Because of the size of the program files used at I4-Tenex, and results files produced by programs run on Illiac IV, arrangements were made to secure additional filestore space; this arrangement enabled them to avoid problems previously encountered in the running of jobs.

The Illiac IV computer was used to perform some three-dimensional supersonic flow field calculations, in which non-linear systems of partial differential equations had to be solved. Illiac IV was found to be between 14 and 20 times faster than the IBM 370/195 computer at Chilton. In addition to establishing just how quickly Illiac IV can produce solutions for complicated systems of hyperbolic PDE's, a technique for programming Illiac IV was devised so that, for the particular class of problems in which they are interested, all that the user has to do is supply a simple routine in the form identical to FORTRAN for a serial machine. The technique removes the need for the user to take account of all the complications associated with intricate data structure for parallel computers like Illiac IV.

Thames Polytechnic

Crowe of the Systems Analysis Division conducted two projects during the year.

(1) The use of the INGRES database at the Rand Corporation to study the relational approach to database system design.

(2) The use of Data Computer America to study a hierarchical structured database.

The first part of the INGRES project was completed and a database was successfully established. The Data Computer America usage has been delayed due to staffing problems.

IX PUBLICATIONS

During 1977, the members of the INDRA group presented many contributions at technical meetings, published a number of technical papers in the Scientific Press, and wrote many Working Notes and Technical Reports. The Technical Reports were either summarised in the papers or conference proceedings. or are referred to in the References. In this section we list only the published papers:

Bennett, C.J. and A.J. Hinchley : "Measurement of the Transmission Control Programme", Conf. on Network Protocols, Liege, Paper G1, 1978.

Higginson, P.L. : "A view of EURONET" the 4th European Network User Workshop, Berlin, 1977 (in the press).

Higginson, P.L. : " A Survey of Methods of Connecting Computer Systems to Packet Switched Networks", Data Communications Networks, London, ONLINE, pp 233-241, 1977.

Higginson, P.L. : "EURONET Terminal Protocols : asynchronous terminal support for EURONET", Commission of the European Communities, 1977.

Higginson, P.L. and Z.Z. Fisher : "Experience with the Initial EPSS Service", EUROCOMP 78, London, pp 581-600, 1978.

Kirstein, P.T. : "Management Questions in relationship to the University College London node of the ARPA Computer Network, ICCC 1976, Toronto, pp 279-285, 1976 (omitted from previous annual report).

Kirstein, P.T. : "Constraint on the development of Services on Public Data Networks", EUSIDIC 1977, Berlin, 1977, (in the press).

Kirstein, P.T. : "Computer Networks", Electronics and Power, November/December, pp 933-939, 1977.

Kirstein, P.T. : "Choice of Communication Media for transmission of facsimile information", Computer Networks, 1978, (in the press).

Kirstein, P.T. : "Facsimile Transmission on Data Networks", Collog. on Facsimile, IEEE, paper 12, 1978.

Kirstein, P.T. and S. Yilmaz : "Facsimile Transmission in Message Processing Systems with Data Management", ICCC 1978 Kyoto, 1978 (in the press).

Kent, S. : "The Automation of Network Access Procedures by names of a High Level Control Language", Ph.D. Thesis, University of London. 1977.

Treadwell, S.W., A.J. Hinchley and C.J. Bennett, "A High Level Network Measurement Tool", EUROCOMP 78, London, pp 35-49, 1978.

Yilmaz, S. and P.T. Kirstein, "UCL Experiments in Facsimile transmission using data base management facilities on ARPANET", EUROCOMP 78, London, pp 789-820, 1978.

X CONCLUSIONS

As last year, a succinct set of conclusions cannot be drawn from the wide range of projects treated in this report. We will summarise, however, some of the conclusions which can be drawn, and indicate the chapter(s) where the work mentioned is described.

(i) We have reached the point with our PDP 9 systems where each extension is taking up a disproportionate amount of effort. We are attempting to connect too many systems in one or two computers which are too small for the job. (2)

(ii) Both the datagram and the virtual call type of network connection gateways can cause problems. In the virtual call types, with host level gateways, the mapping of protocols may be difficult; however the two networks can use quite different philosophies and protocols. With datagram gateways, at the packet level, host-host flow control and sequencing protocols have to be agreed to run on an end-to-end basis between the collaborating Hosts. (2,3,4,5).

(iii) Mapping protocols at all levels between virtual call networks by Host level gateways is fairly straight forward. However certain features, e.g. echoing of characters, may not be resolvable in the Gateway; Bilateral Source Host - Destination Host agreement may be required for a fully satisfactory service. (3,4)

(iv) EPSS is becoming an operational entity - but Hosts are slow to implement the High Level Protocols. (3)

(v) There is a broad scope of work on the X25 protocol. Much work can be done by making EPSS seem like an X25 network. This transformation by a Front-end Processor is quite feasible.(4)

(vi) Care must be taken to reduce the size of X25 implementations. Early versions written in a high level language are large; presumably assembler versions will be much smaller. (4)

(vii) Traffic generation and collection, with time-stamped packets, is a very powerful method of analysing the performance of packet switched networks. (5)

(viii) The integration of facsimile techniques with message processing is very promising.
With a digital facsimile device, good message processors and cheap archival storage, a very attractive offering may emerge.(7)

(ix) Considerable attention should be paid to reducing the cost of processing in archival systems like the Terabit store. Bulk rates for the transmission of large quantities of data are very desirable. (7)

(x) The type of use made of the UCL node of ARPANET is becoming more complex; however a steady population of researchers from many disciplines, are integrating the use of the link into their research. (8)

REFERENCES

Unless stated otherwise, all references are to documents produced by the INDRA Group of the Department of Statistics and Computer Science of University College London.

BENNETT 1977 : Bennett, C.J., Edge, S.W., and A.J. Hinchley, "Issues in the Interconnection of Datagram Networks", INDRA Note 637, 1977.

HOLMES 1977 : Holmes, P.L., "On-Line Information Retrieval - an Introduction to the British Library's Short Term Experimental Information Network Project", Vol. II Experimental Use of Medical Information Services, British Library Research and Development Department. ISBN-0-905984-10-2, 1977

INDRA 1975, "Collected Papers on the Experience with the London node of the ARPA computer Network, TR22, 1975.

KENT 1977 : Kent, S., "The Automation of Network Access Procedures by means of a HIgh Level Control Language", Ph.D. Thesis University of London, 1977.

KIRSTEIN 1977 : Kirstein, P.T., "University College London ARPANET Project, Annual Report 1 January 1976-31 December 1976, TR34 1977.

KIRSTEIN 1978 : Kirstein, P.T., "Choice of Communication Media for Transmission of Facsimile Information", Computer Networks (in the Press).

STOKES 1977 A : Stokes, A.V., "The INDRA Network Measurement Project", TR 30, 1977.

STOKES 1977 B : Stokes, A.V., "Measurement of PDP9A Port Usage, January - June 1977", TR 42, 1977.

STOKES 1977 C : Stokes, A.V. "Access to an Information Retrieval System via a Distributed Computer Network", TR 40, 1977.

TREADWELL 1977 : Treadwell, S., "The Measured Characteristics of Traffic in the UCL node of the ARPA Network", TR 33, 1977.

(When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TR - 48	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) University College London ARPANET Project Annual Report		5. TYPE OF REPORT & PERIOD COVERED ANNUAL, JAN-DECEMBER 1977 6. PERFORMING ORG. REPORT NUMBER TR - 48
7. AUTHOR(s) Professor Peter T. Kirstein		8. CONTRACT OR GRANT NUMBER(s) N-00014-77-G-005
9. PERFORMING ORGANIZATION NAME AND ADDRESS University College London, Gower Street, London WC1E 6BT		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defence Advanced Research Projects Agency 1400 Wilson Boulevard, Arlington, Virginia 22209		12. REPORT DATE APRIL, 1978 13. NUMBER OF PAGES 156
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE U
16. DISTRIBUTION STATEMENT (of this Report) No.1 APPROVED FOR PUBLIC RELEASE DISTRIBUTION UNLIMITED		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Measurements of Network Usage, Packet Switched Networks, ARPANET, SATNET, EPSS, Digital Facsimile, Internetworking		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report describes the work of the UCL INDRA group during 1977. The bulk of the report consists of papers which have been published or are in the press. These cover the group's activities with the UK Experimental Packet Switched Service, with measurement tools for a Packet Satellite Network, with measurement of a specific Host-Host Protocol, and with the combination of Facsimile Techniques with Message Processing		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

in packet-switched data networks. There are also short summaries of the UCL work on the X25 Network Access Protocols and on the interconnection of computer networks. Finally, the current usage being made of the UCL node of ARPANET is analysed.